Matematica delle Reti Neurali

Volume IX

AI Agentiva e Multi-Agente · Inference-Time Compute
Generazione Video · Continual Learning · Topologia Applicata

Silvano Salvador

Continuazione dei Volumi I–VIII (Sezioni 0–510)

Sezioni 511–542 · Derivazioni complete · Ogni simbolo definito

Temi trattati: ReAct · Mean-Field Games · MARL · Nash Equilibria · MCTS per LLM · Process Reward Models · Best-of-N con Verifier · GRPO · Video Diffusion · Attenzione Spaziotemporale · EWC · Progressive Networks · GEM · A-GEM · Omologia Persistente · Mapper Algorithm · Betti Numbers dei Loss Landscape · Grokking topologico

Prefazione al Volume IX

Gli otto volumi precedenti di questa serie hanno costruito un panorama matematico di straordinaria ampiezza: dai fondamenti del neurone artificiale (Vol. I) fino alla fisica statistica dei trasformatori, all'equivarianza SE(3) nelle reti per proteine, alle leggi di scala Chinchilla e alla biologia computazionale profonda (Vol. VIII). Eppure alcune aree fondamentali—aree con un corpus matematico rigoroso, in rapida crescita e di primaria rilevanza pratica—erano rimaste sistematicamente escluse dal ciclo.

Questo volume le affronta con la stessa ambizione dei precedenti: ogni concetto è fondato su definizioni formali, ogni architettura è smontata nei suoi componenti elementari, ogni algoritmo è derivato da principi primi. L'obiettivo è che un lettore con la preparazione del Volume I riesca a seguire ogni passo senza lacune.

Sezione A (§511–§518): *AI Agentiva e Multi-Agente.* La matematica dei sistemi agentici—dal framework ReAct alla teoria dei giochi multi-agente—è forse la lacuna più rilevante del ciclo, data la centralità attuale degli LLM come agenti autonomi. Si derivano: il ciclo osserva-pensa-agisci come MDP parzialmente osservabile; il tool-use formale; il Multi-Agent Reinforcement Learning (MARL) con gli algoritmi IQL, QMIX e MAPPO; gli equilibri di Nash e lo Shapley value; i giochi a potenziale con garanzie di convergenza; e—novità rispetto alla versione precedente—i *mean-field games* come limite termodinamico del MARL a molti agenti, insieme all'analisi degli *equilibri correlati* che generalizzano Nash in modo computazionalmente trattabile.

Sezione B (§519–§524): *Inference-Time Compute Scaling.* Il ciclo ha trattato le scaling laws di Kaplan e Chinchilla (Vol. II), ma non il test-time compute scaling. Si trattano: MCTS applicato agli LLM con analisi del regret UCB1; i Process Reward Models (PRM) e la loro derivazione come stima della funzione valore; il framework best-of-N con verifier e la sua curva di scaling; la beam search come caso speciale di MCTS; e i modelli *thinking* (o1, Gemini Flash Thinking, QwQ) con l'algoritmo GRPO e la sua derivazione dal principio di ottimizzazione di politica vincolata. La sezione include ora l'analisi dell'utilità marginale del compute e il collegamento formale con il framework POMDP della Sezione A.

Sezione C (§525–§529): *Generazione Video e Modelli Spaziotemporali.* I modelli di generazione video richiedono estensioni non banali dell'architettura diffusion. Si derivano: il Video Diffusion Model (Ho et al., 2022) con attenzione spaziotemporale fattorizzata; il 3D-VAE con loss completa; l'architettura Sora basata su Diffusion Transformer (DiT) spaziotemporale con RoPE 3D; VideoGPT come modello autoregressivo su token video; e i Consistency Models per la generazione video in pochi passi. La sezione include ora la derivazione del *flow matching* per video e l'analisi della complessità computazionale comparata tra le diverse architetture.

Sezione D (§530–§534): *Continual e Lifelong Learning.* La matematica del catastrophic forgetting ha basi teoriche precise spesso trascurate. Si derivano: la geometria del forgetting tramite approssimazione quadratica della loss; la teoria PAC su distribuzioni non stazionarie con concept drift brusco e graduale; l'Elastic Weight Consolidation (EWC) come regolarizzazione bayesiana con matrice di Fisher; le Progressive Neural

Networks con analisi del transfer; il Gradient Episodic Memory (GEM) con garanzia di non-peggioramento; e—ampliamento rispetto alla versione precedente—A-GEM (Averaged GEM) con la sua complessità ridotta e la Synaptic Intelligence (SI) come alternativa online a EWC.

Sezione E (§535–§542): *Topologia Applicata al Deep Learning.* Si costruisce l'omologia simpliciale da zero, si dimostra il Lemma Fondamentale $\partial_{k-1}\partial_k = 0$, si enuncia e motiva il Teorema di Struttura della Persistent Homology, si stabilisce la stabilità tramite distanza bottleneck, e si applicano questi strumenti ai loss landscape, alle rappresentazioni neurali e al fenomeno del grokking. Nuovi sviluppi: il *diagramma di persistenza come vettore* (persistenza vettoriale) per applicazioni di machine learning; e l'analisi topologica delle *trasformazioni di fase* durante l'addestramento.

Nota per il lettore. Ogni sezione è autocontenuta: tutti i simboli vengono ridefiniti localmente. I rimandi ai volumi precedenti sono indicati esplicitamente nella forma (Vol. N, §M). Le dimostrazioni complete terminano con □ e possono essere saltate in prima lettura senza perdere il filo della trattazione.

Indice

Prefazione **3**

1 Sezione A — AI Agentiva e Multi-Agente **7**
1.1 Agenti LLM come POMDP: il framework ReAct 7
1.2 Multi-Agent Reinforcement Learning 10

2 Sezione B — Inference-Time Compute Scaling **17**
2.1 Test-Time Compute: Motivazione e Scaling Laws 17

3 Sezione C — Generazione Video e Modelli Spaziotemporali **23**
3.1 Formalizzazione del Problema: Video come Sequenza 4D 23

4 Sezione D — Continual e Lifelong Learning **29**
4.1 Catastrophic Forgetting: Analisi Teorica 29

5 Sezione E — Topologia Applicata al Deep Learning **35**
5.1 Fondamenti di Omologia Simpliciale 35

6 Approfondimenti e Connessioni Trasversali **43**
6.1 Teoria dell'Informazione nei Sistemi Agentici 43
6.2 Analisi Spettrale dei Modelli di Generazione Video 44
6.3 Analisi di Complessità degli Algoritmi di Continual Learning 45
6.4 Applicazioni della TDA alle Architetture Transformer 46
6.5 Scaling Laws Unificate: Training-Time e Test-Time 47
6.6 Prove di Convergenza Estese 48

6.7 Esempi Computazionali Espliciti 49
6.8 Connessioni tra le Sezioni del Volume 50

Conclusioni e Prospettive **53**

7 Derivazioni Avanzate e Metodi di Secondo Ordine **55**
7.1 Ottimizzazione Naturale del Gradiente nel MARL 55
7.2 Teoria delle Approssimazioni Funzionali per il POMDP 56
7.3 Modelli di Diffusione: Connessioni Profonde 57
7.4 Metodi Kernel per la Persistent Homology 58
7.5 Teoria della Generalizzazione per Modelli Diffusivi 59
7.6 Continual Learning con Ottimizzazione Distribuita 60

8 Fondamenti Probabilistici e Teoria della Misura **63**
8.1 Processi Stocastici e Martingale 63
8.2 Trasporto Ottimale e Distanze tra Distribuzioni 64
8.3 Teoria della Misura e Topologia 65
8.4 Geometria dell'Informazione e Reti Neurali 66
8.5 Analisi Funzionale e Spazi di Funzioni Neurali 67
8.6 Algoritmi di Campionamento per il Test-Time Compute 68

9 Derivazioni Complete: dalle Equazioni ai Codici **71**
9.1 Derivazione Completa di GRPO da Principi Primi 71
9.2 Derivazione Completa di EWC: dalla Fisher all'Algoritmo 72
9.3 Connessioni con la Teoria dei Sistemi Dinamici 74
9.4 Numerical Analysis dei Solver per MCTS 75
9.5 Reti di Attenzione e Geometria Riemanniana 75
9.6 Formulazione Variazionale del Continual Learning 77

10 Reinforcement Learning Avanzato per Sistemi Agentici **79**
10.1 Reward Shaping e Potential-Based Shaping 79
10.2 Teoria degli Algoritmi Evolutivi per MARL 80
10.3 Teoria dei Grafi Computazionali per l'Addestramento 81
10.4 Analisi di Stabilità dei Video Diffusion Models 82
10.5 Teoria dell'Informazione Topologica 83
10.6 Algoritmi di Ottimizzazione di Secondo Ordine 84
10.7 Analisi degli Errori nelle Pipeline di Generazione Video 85
10.8 Connessioni con la Fisica Statistica 87

11 Teoria Avanzata dell'Ottimizzazione per il Deep Learning **89**
11.1 Loss Landscape: Geometria e Ottimizzazione 89
11.2 Convergenza degli Algoritmi di Generazione: Analisi Quantitativa . . . 91
11.3 Analisi di Complessità Computazionale dei Modelli Trattati 92
11.4 Metodi Variazionali e Approssimazioni 93
11.5 Metriche di Valutazione per il Continual Learning 94
11.6 Reti Neurali su Grafi e Topologia 95

12 Implementazione e Considerazioni Pratiche **97**
12.1 Implementazione di GRPO in Pseudo-Codice 97

12.2 Implementazione di EWC per Transformer 98
12.3 Considerazioni sull'Hardware e la Scalabilità 99

13 Esercizi e Problemi Aperti **101**
13.1 Esercizi con Soluzione 101
13.2 Problemi Aperti 103

Indice delle Sezioni **105**

Glossario dei Simboli Principali **107**

Bibliografia **109**

Capitolo 1

Sezione A — AI Agentiva e Multi-Agente

1.1 Agenti LLM come POMDP: il framework ReAct

§511 – Processi di Decisione Markoviani Parzialmente Osservabili

Un agente basato su LLM percepisce il mondo attraverso testo e agisce generando testo: questa struttura si formalizza naturalmente come *Partially Observable Markov Decision Process* (POMDP).

Definizione 1.1 (POMDP). Un POMDP è una tupla $M = (S, A, O, T, Z, R, \gamma)$ dove:

- S è lo spazio degli stati (non osservabile direttamente);
- A è lo spazio delle azioni (per un agente LLM: testo generabile);
- O è lo spazio delle osservazioni (testo ricevuto, risultati tool);
- $T(s' \mid s, a) = P(s_{t+1} = s' \mid s_t = s, a_t = a)$ è la funzione di transizione;
- $Z(o \mid s', a) = P(o_t = o \mid s_t = s', a_{t-1} = a)$ è la funzione di emissione;
- $R : S \times A \to \mathbb{R}$ è la funzione di ricompensa;
- $\gamma \in [0, 1)$ è il fattore di sconto.

La differenza cruciale rispetto a un MDP (Vol. III, §109–116) è che l'agente non osserva s_t direttamente, ma solo $o_t \in O$. Deve quindi mantenere una *belief state* $b_t \in \Delta(S)$ (distribuzione sugli stati), aggiornata tramite il teorema di Bayes:

$$b_{t+1}(s') = \frac{Z(o_{t+1} \mid s', a_t) \sum_s T(s' \mid s, a_t)\, b_t(s)}{\sum_{s''} Z(o_{t+1} \mid s'', a_t) \sum_s T(s'' \mid s, a_t)\, b_t(s)}. \tag{1.1}$$

Per un agente LLM lo stato latente s_t comprende l'intera storia della conversazione e lo stato del mondo esterno (file system, database, output dei tool). La belief state è implicitamente rappresentata nel contesto del modello.

Nota. Calcolare esattamente (1.1) richiede integrare su S, che per applicazioni reali è continuo o esponenzialmente grande. Gli LLM approssimano questa computazione tramite generazione autoregressiva, usando il contesto come proxy della belief state. La dimensione del contesto determina quindi la *capacità di memoria* effettiva dell'agente.

Funzione valore per il POMDP

La funzione valore ottimale nel POMDP è definita sullo spazio delle belief state $b \in \Delta(S)$, non sullo spazio degli stati S. Questo è cruciale: il POMDP è un MDP sullo spazio $\Delta(S)$, che è continuo anche quando S è finito.

Definizione 1.2 (Valore del POMDP)**.** La funzione valore ottimale è:

$$V^*(b) = \max_{a \in A} \left[\sum_s b(s)\, R(s,a) + \gamma \sum_{o \in O} P(o \mid b, a)\, V^*(b_{a,o}) \right], \tag{1.2}$$

dove $b_{a,o}$ è la belief state aggiornata dopo azione a e osservazione o, e $P(o \mid b, a) = \sum_{s'} Z(o \mid s', a) \sum_s T(s' \mid s, a) b(s)$.

Teorema 1.1 (Lineare a tratti di V^*)**.** La funzione V^* è concava e lineare a tratti su $\Delta(S)$: esiste un insieme finito di vettori $\{\alpha_j\} \subset \mathbb{R}^{|S|}$ (*piani* α) tale che

$$V^*(b) = \max_j \sum_s b(s)\, \alpha_j(s). \tag{1.3}$$

Dimostrazione. La base è $V_0^*(b) = \max_a \sum_s b(s) R(s,a)$, che è un massimo di lineari in b, dunque lineare a tratti. L'operatore di Bellman preserva questa struttura ad ogni iterazione per il teorema di Smallwood e Sondik (1973): ogni iterazione produce un nuovo insieme finito di piani α. Al limite, la struttura è mantenuta poiché $\Delta(S)$ è compatto e V^* è il punto fisso dell'operatore di Bellman, che è una contrazione per $\gamma < 1$. □

§512 – ReAct: Ragionamento e Azione Interlacciati

Il framework ReAct (Yao et al., 2023) formalizza il ciclo Reasoning + Acting come sequenza di passi:

$$\tau = (o_1, r_1, a_1, o_2, r_2, a_2, \ldots, o_T) \tag{1.4}$$

dove o_t è l'osservazione al passo t, r_t è una catena di pensiero (*thought*) generata dall'LLM, e a_t è un'azione concreta.

Definizione 1.3 (Politica ReAct)**.** Sia π_θ un LLM con parametri θ e finestra di contesto $c_t = (o_1, r_1, a_1, \ldots, o_t)$. La politica ReAct è:

$$r_t \sim \pi_\theta(\cdot \mid c_t, \text{“Pensiero:”}), \qquad a_t \sim \pi_\theta(\cdot \mid c_t, r_t, \text{“Azione:”}). \tag{1.5}$$

Il pensiero r_t non modifica l'ambiente ma condiziona la generazione di a_t: è un mezzo di pianificazione implicita.

Analisi formale dell'efficacia del reasoning

Sia $H(X)$ l'entropia di Shannon della variabile aleatoria X. Poiché r_t è informazione aggiuntiva sul contesto:

$$H(a_t \mid c_t, r_t) \leq H(a_t \mid c_t). \tag{1.6}$$

La catena di pensiero riduce l'incertezza dell'azione. Tuttavia, la riduzione è nulla se r_t è indipendente da a_t dato c_t: il beneficio del reasoning è condizionato alla *rilevanza* del pensiero.

Proposizione 1.1 (Monotonicità del valore con il reasoning)**.** Sia π^{ReAct} la politica che genera pensiero prima di agire e π^{direct} la politica che agisce direttamente. Se r_t è calcolato ottimalmente (ovvero $r_t^* = \arg\max_r \mathbb{E}_{a\sim\pi(\cdot|c,r)}[V^*(b_{t+1})]$), allora $V^{\pi^{\text{ReAct}}}(b) \geq V^{\pi^{\text{direct}}}(b)$.

Dimostrazione. Sia $a^* = \arg\max_a \mathbb{E}_{s'\sim T(\cdot|s,a)}[R(s,a) + \gamma V^*(b')]$ l'azione Bellman-ottimale. La politica ReAct sceglie r_t per massimizzare $\mathbb{E}[V^*(b')]$, quindi trova $r^* = \arg\max_r P(a = a^* \mid c_t, r) \cdot \mathbb{E}[V^*(b') \mid a^*]$. Poiché la distribuzione marginale su a soddisfa $\mathbb{E}_{a|c}[\cdot] = \mathbb{E}_{r|c}\mathbb{E}_{a|c,r}[\cdot]$, e la politica diretta corrisponde a integrare su r senza ottimizzare, la politica ReAct ottimizza esplicitamente su r e non può fare peggio. □

Analisi del regret multi-passo

Definiamo il *regret cumulativo* di un agente ReAct su un orizzonte di T passi:

$$\text{Regret}(T) = \sum_{t=1}^{T} \left[V^*(b_t) - V^{\pi^{\text{ReAct}}}(b_t) \right]. \tag{1.7}$$

Quando il pensiero è generato da un LLM imperfetto (non ottimale), il regret dipende dall'*errore di approssimazione* del reasoning. Formalmente, sia $\epsilon_r = \max_t \mathbb{E}[\|r_t - r_t^*\|]$ l'errore medio del reasoning. Allora:

$$\text{Regret}(T) \leq \frac{T \cdot L_V \cdot \epsilon_r}{1 - \gamma}, \tag{1.8}$$

dove L_V è la costante di Lipschitz di V^* rispetto alle azioni indotte dal pensiero.

§513 – Tool-Use Formale: Funzioni Esterne come Estensione dell'Azione

Nel framework agentivo, le azioni $a_t \in A$ includono chiamate a funzioni esterne (strumenti):

$$A = A^{\text{testo}} \cup A^{\text{tool}}, \qquad A^{\text{tool}} = \bigsqcup_{k=1}^{K} \{f_k\} \times X_k \tag{1.9}$$

dove $f_k : X_k \to Y_k$ è il k-esimo strumento con dominio X_k e codominio Y_k.

Definizione 1.4 (Tool-Call come Token Speciale)**.** Si estende il vocabolario V con token speciali:

$$V^+ = V \cup \{\texttt{<tool_call>}, \texttt{</tool_call>}, \texttt{<tool_result>}, \texttt{</tool_result>}\}.$$

La probabilità di generare una tool-call corretta si fattorizza come:

$$P(a_t = f_k(x) \mid c_t) = \underbrace{\prod_{v \in \text{nome}(f_k)} \pi_\theta(v \mid c_t)}_{\text{selezione tool}} \cdot \underbrace{\prod_{i=1}^{|x|} \pi_\theta(x_i \mid c_t, f_k, x_{<i})}_{\text{argomenti}}. \tag{1.10}$$

Fine-tuning per il tool-use

Il fine-tuning massimizza il log-likelihood di traiettorie (c_t, a_t^*) dove a_t^* è la tool-call corretta annotata da esperti:

$$\mathcal{L}_{\text{tool}}(\theta) = -\mathbb{E}_{(c_t, a_t^*) \sim \mathcal{D}} \left[\log P_\theta(a_t^* \mid c_t) \right]. \tag{1.11}$$

Composizione di strumenti e pianificazione

Un caso importante è la *composizione* di strumenti: l'output di f_k diventa input di f_j. Formalmente, un *piano* è una sequenza di tool-call $\mathbf{a} = (a_1, a_2, \ldots, a_m)$ tale che $Y_{k_i} \subseteq X_{k_{i+1}}$ per ogni i. La probabilità di un piano corretto è:

$$P(\mathbf{a} \mid c_0) = \prod_{i=1}^{m} P(a_i \mid c_0, a_1, \ldots, a_{i-1}, o_1, \ldots, o_{i-1}), \tag{1.12}$$

dove $o_i = f_{k_i}(x_i)$ è l'output del i-esimo strumento inserito nel contesto.

1.2 Multi-Agent Reinforcement Learning

§514 – Giochi a Somma Generale: Definizioni e Notazione

Definizione 1.5 (Gioco a N Agenti)**.** Un gioco a N agenti è una tupla $\mathcal{G} = (\mathcal{N}, S, \{A_i\}_{i \in \mathcal{N}}, T, \{R_i\}_{i \in \mathcal{N}}, \gamma)$ dove:

- $\mathcal{N} = \{1, \ldots, N\}$ è l'insieme degli agenti;
- S è lo spazio degli stati condiviso;
- A_i è lo spazio delle azioni dell'agente i;
- $A = \times_{i=1}^{N} A_i$ è lo spazio delle azioni congiunte;
- $T : S \times A \to \Delta(S)$ è la transizione;
- $R_i : S \times A \to \mathbb{R}$ è la ricompensa dell'agente i;
- $\gamma \in [0, 1)$ è il fattore di sconto comune.

Se $\sum_i R_i = 0$ per ogni (s, a) il gioco è a *somma zero*. Se $R_i = R_j$ per ogni i, j è *cooperativo*.

Giochi estensivi e forma normale

Un *gioco in forma estesa* aggiunge alla tupla un albero di gioco $\mathcal{T} = (V, E, u)$ dove ogni foglia $v \in V$ è un'uscita terminale con ricompense $R_i(v)$. La forma normale si ottiene enumerando le *strategie pure* s_i (piani completi per ogni nodo di informazione) e costruendo la matrice di payoff $U_i(s_1, \ldots, s_N)$.

Definizione 1.6 (Nodo di Informazione)**.** Un nodo di informazione I_i per il giocatore i è un insieme di nodi del gioco che i non riesce a distinguere (per informazione imperfetta). La strategia di i è una mappa $\sigma_i : I_i \to \Delta(A_i(I_i))$, dove $A_i(I_i)$ sono le azioni disponibili in I_i.

§515 – Equilibrio di Nash: Esistenza e Caratterizzazione

Definizione 1.7 (Politica Mista e Equilibrio di Nash)**.** Una politica mista per l'agente i è $\pi_i \in \Delta(A_i)$. Un profilo di politiche $\pi^* = (\pi_1^*, \dots, \pi_N^*)$ è un *equilibrio di Nash* se per ogni agente i e ogni deviazione π_i:

$$V_i(\pi_i^*, \pi_{-i}^*) \geq V_i(\pi_i, \pi_{-i}^*) \tag{1.13}$$

dove $\pi_{-i}^* = (\pi_j^*)_{j \neq i}$ e $V_i(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_t)\right]$.

Teorema 1.2 (Nash, 1951)**.** Ogni gioco finito a N giocatori con spazi di azioni finiti ammette almeno un equilibrio di Nash in politiche miste.

Dimostrazione. Definiamo la best-response dell'agente i come $\mathrm{BR}_i(\pi_{-i}) = \arg\max_{\pi_i \in \Delta(A_i)} V_i(\pi_i, \pi_{-i})$. Poiché V_i è lineare in π_i, il massimo di una funzione lineare su un simplesso è attinto su un insieme convesso non vuoto. La mappa $\mathrm{BR} : \Delta(A) \to 2^{\Delta(A)}$, definita da $\mathrm{BR}(\pi) = \prod_i \mathrm{BR}_i(\pi_{-i})$, è upper-hemicontinua e ha valori convessi non vuoti in un compatto convesso $\Delta(A)$. Per il *teorema del punto fisso di Kakutani* esiste $\pi^* \in \mathrm{BR}(\pi^*)$, ovvero un equilibrio di Nash. □

Equilibri correlati

Definizione 1.8 (Equilibrio Correlato (Aumann, 1987))**.** Una distribuzione $\sigma \in \Delta(A)$ sulle azioni congiunte è un *equilibrio correlato* se per ogni giocatore i e ogni deviazione $d_i : A_i \to A_i$:

$$\sum_{a \in A} \sigma(a)\, R_i(a) \geq \sum_{a \in A} \sigma(a)\, R_i(d_i(a_i), a_{-i}). \tag{1.14}$$

Proposizione 1.2 (Inclusione degli equilibri)**.** Ogni equilibrio di Nash (in politiche miste) è un equilibrio correlato, ma non viceversa. L'insieme degli equilibri correlati è un politopo convesso, quindi computazionalmente trattabile in modo lineare.

La struttura convessa degli equilibri correlati è cruciale per il MARL: algoritmi come *regret matching* (Hart e Mas-Colell, 2000) convergono esattamente all'insieme degli equilibri correlati, pur non richiedendo conoscenza delle funzioni di payoff degli altri agenti.

§516 – Algoritmi MARL: IQL, QMIX e MAPPO

Independent Q-Learning (IQL)

La soluzione più semplice al MARL: ogni agente i addestra una rete $Q_i(s, a_i; \theta_i)$ con la loss DQN standard (Vol. III, §111), trattando gli altri agenti come parte dell'ambiente:

$$\mathcal{L}_i(\theta_i) = \mathbb{E}\left[\left(R_i + \gamma \max_{a_i'} Q_i(s', a_i'; \bar{\theta}_i) - Q_i(s, a_i; \theta_i)\right)^2\right]. \tag{1.15}$$

IQL non converge in generale in ambienti non stazionari perché, dal punto di vista di ciascun agente, gli altri agenti cambiano la loro politica durante l'addestramento, violando l'ipotesi di stazionarietà dell'ambiente. Tuttavia è computazionalmente scalabile: $O(N)$ reti indipendenti.

QMIX (Rashid et al., 2018): monotone value factorization

Definizione 1.9 (QMIX Factorization)**.** Il valore congiunto Q_{tot} è una funzione monotona non decrescente dei valori individuali Q_i:

$$Q_{\text{tot}}(s, \mathbf{a}) = f_s(Q_1(s_1, a_1), \ldots, Q_N(s_N, a_N)) \tag{1.16}$$

dove f_s è una rete ipernetwork condizionata su s con pesi non negativi, garantendo $\partial Q_{\text{tot}} / \partial Q_i \geq 0$ per ogni i.

Teorema 1.3 (Individual-Global-Max (IGM))**.** Sotto la fattorizzazione QMIX vale il principio IGM:

$$\arg\max_{\mathbf{a} \in A} Q_{\text{tot}}(s, \mathbf{a}) = \left(\arg\max_{a_i \in A_i} Q_i(s_i, a_i)\right)_{i=1}^{N}. \tag{1.17}$$

Dimostrazione. Poiché f_s è non decrescente in ogni Q_i, massimizzare f_s rispetto all'azione congiunta $\mathbf{a}$ equivale a massimizzare ciascun Q_i separatamente rispetto ad a_i, con gli altri fissati. Formalmente, per ogni i e ogni a_{-i}:

$$a_i^* = \arg\max_{a_i} Q_{\text{tot}}(s, (a_i, a_{-i})) = \arg\max_{a_i} f_s(\ldots, Q_i(s_i, a_i), \ldots) = \arg\max_{a_i} Q_i(s_i, a_i)$$

dove l'ultima uguaglianza segue dalla monotonicità di f_s in Q_i. Poiché questo vale per ogni i simultaneamente, il principio IGM è stabilito. □

Multi-Agent PPO (MAPPO)

Per politiche continue, MAPPO estende PPO (Vol. III, §115) al caso multi-agente con critici centralizzati (paradigma CTDE: Centralised Training, Decentralised Execution):

$$\mathcal{L}_i^{\text{CLIP}}(\theta_i) = \mathbb{E}_t\left[\min\Big(r_t^i(\theta_i)\,\hat{A}_t^i,\ \text{clip}(r_t^i(\theta_i), 1-\varepsilon, 1+\varepsilon)\,\hat{A}_t^i\Big)\right], \tag{1.18}$$

$$r_t^i(\theta_i) = \frac{\pi_{\theta_i}(a_t^i \mid o_t^i)}{\pi_{\theta_i^{\text{old}}}(a_t^i \mid o_t^i)}, \tag{1.19}$$

$$\hat{A}_t^i = \sum_{l=0}^{\infty} (\gamma\lambda)^l\,\delta_{t+l}^i, \qquad \delta_t^i = R_i(s_t, \mathbf{a}_t) + \gamma V_\phi^i(s_{t+1}) - V_\phi^i(s_t). \tag{1.20}$$

Il critico $V_\phi^i(s_t)$ utilizza lo stato globale s_t, che include le osservazioni di tutti gli agenti. Durante l'esecuzione, ciascun agente usa solo o_t^i (esecuzione decentralizzata).

§517 – Teoria dei Giochi Cooperativa: Shapley Value e Credito

In sistemi multi-agente cooperativi, il *problema dell'attribuzione del credito* chiede: quanto ha contribuito ciascun agente al successo collettivo?

Definizione 1.10 (Shapley Value)**.** Dato un gioco di coalizione $(\mathcal{N}, v)$ con $v : 2^{\mathcal{N}} \to \mathbb{R}$, lo *Shapley value* dell'agente i è:

$$\phi_i(v) = \sum_{S \subseteq \mathcal{N}\setminus\{i\}} \frac{|S|!\,(N - |S| - 1)!}{N!}\left[v(S \cup \{i\}) - v(S)\right]. \tag{1.21}$$

Teorema 1.4 (Unicità dello Shapley value (Shapley, 1953))**.** Lo Shapley value è l'unica funzione $\phi : \mathbb{R}^{2^N} \to \mathbb{R}^N$ che soddisfa simultaneamente:

1. **Efficienza:** $\sum_i \phi_i(v) = v(\mathcal{N})$;
2. **Simmetria:** se $v(S \cup \{i\}) = v(S \cup \{j\})$ per ogni S, allora $\phi_i(v) = \phi_j(v)$;
3. **Dummy:** se $v(S \cup \{i\}) = v(S)$ per ogni S, allora $\phi_i(v) = 0$;
4. **Additività:** $\phi_i(v + w) = \phi_i(v) + \phi_i(w)$.

Dimostrazione. [Sketch] L'esistenza si verifica per sostituzione diretta. Per l'unicità: i quattro assiomi determinano univocamente ϕ su ogni gioco *unanimità* u_T (dove $u_T(S) = 1$ se $T \subseteq S$, 0 altrimenti). Poiché i giochi unanimità formano una base dello spazio dei giochi di coalizione, e ϕ è additivo, è univocamente determinato su tutto lo spazio. □

Shapley value nel MARL

In MARL, si definisce $v(S)$ come il valore atteso del ritorno ottenuto dalla coalizione S giocando ottimalmente (con gli agenti non in S che seguono una politica di riferimento π^0):

$$v(S) = \mathbb{E}_{\pi_S^*, \pi^0_{\mathcal{N}\setminus S}}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \mathbf{a}_t)\right]. \tag{1.22}$$

Il costo computazionale dello Shapley esatto è $O(2^N)$: esponenziale in N. Per sistemi con molti agenti si usano metodi Monte Carlo:

$$\hat{\phi}_i(v) = \frac{1}{M}\sum_{m=1}^{M}\left[v(\hat{S}_m \cup \{i\}) - v(\hat{S}_m)\right], \tag{1.23}$$

dove $\hat{S}_m$ è un sottoinsieme campionato uniformemente da $2^{\mathcal{N}}$. Con $M = O(N \log N/\varepsilon^2)$ campioni si ottiene un'approssimazione ε-accurata con alta probabilità (Maleki et al., 2013).

§518 – Equilibri di Nash e Convergenza in Giochi a Potenziale

Definizione 1.11 (Gioco a Potenziale Esatto (Monderer e Shapley, 1996))**.** Un gioco è a *potenziale esatto* se esiste $\Phi : S \times A \to \mathbb{R}$ tale che per ogni i, ogni a_{-i}, ogni a_i, a_i':

$$R_i(a_i, a_{-i}) - R_i(a_i', a_{-i}) = \Phi(a_i, a_{-i}) - \Phi(a_i', a_{-i}). \tag{1.24}$$

Teorema 1.5 (Convergenza nei Giochi a Potenziale)**.** In un gioco a potenziale esatto con spazi di azioni finiti, l'algoritmo *best response dynamics* converge a un equilibrio di Nash puro in tempo finito.

Dimostrazione. Ogni aggiornamento incrementa Φ:

$$\Phi(a_i^{\text{new}}, a_{-i}) - \Phi(a_i^{\text{old}}, a_{-i}) = R_i(a_i^{\text{new}}, a_{-i}) - R_i(a_i^{\text{old}}, a_{-i}) > 0$$

se a_i^{new} è una best-response stretta. Poiché Φ è bounded (spazio finito di azioni) e strettamente crescente ad ogni aggiornamento, il processo termina in un punto fisso che è, per costruzione, un equilibrio di Nash. □

Mean-Field Games: il limite termodinamico del MARL

Quando il numero di agenti $N \to \infty$, tracciare le interazioni tra tutte le coppie diventa intrattabile. I *mean-field games* (MFG; Lasry e Lions, 2007; Huang et al., 2006) risolvono questo problema sostituendo le interazioni con il singolo agente con un'interazione con la *distribuzione media degli agenti*.

Definizione 1.12 (Mean-Field Game)**.** Un MFG è un sistema di equazioni accoppiate:

$$\text{(HJB)} \quad -\partial_t V(x,t) + H(x, \nabla_x V, \mu_t) = 0, \tag{1.25}$$

$$\text{(FPK)} \quad \partial_t \mu_t + \nabla_x \cdot (\mu_t \, v^*(x, \nabla_x V, \mu_t)) = 0, \tag{1.26}$$

dove $V(x,t)$ è la funzione valore di un agente rappresentativo in stato x al tempo t, $\mu_t \in \mathcal{P}(\mathbb{R}^d)$ è la distribuzione della popolazione, e $H(x,p,\mu)$ è l'Hamiltoniano.

L'equazione (1.25) è l'equazione di Hamilton-Jacobi-Bellman: descrive la politica ottimale di un singolo agente dato il campo medio μ_t. L'equazione (1.26) è la Fokker-Planck-Kolmogorov: descrive come la distribuzione della popolazione evolve quando tutti gli agenti seguono la politica ottimale. Un MFG *equilibrio* è una coppia (V^*, μ^*) che soddisfa entrambe le equazioni simultaneamente.

Proposizione 1.3 (Limite MFG del MARL a N agenti)**.** Sotto ipotesi di regolarità, l'equilibrio di Nash del gioco a N agenti converge all'equilibrio del MFG quando $N \to \infty$, con tasso di convergenza $O(1/\sqrt{N})$ in distanza Wasserstein.

Capitolo 2

Sezione B — Inference-Time Compute Scaling

2.1 Test-Time Compute: Motivazione e Scaling Laws

§519 – La Dicotomia Training-Time vs Test-Time Compute

Le scaling laws di Kaplan (2020) e Chinchilla (Hoffmann et al., 2022) (Vol. II, §64; Vol. VIII, §451) caratterizzano come la performance migliora con il compute di addestramento. Un filone di ricerca distinto, emerso con i modelli o1 (OpenAI, 2024) e Gemini 2.0 Flash Thinking, studia come la performance migliora allocando più compute al momento dell'inferenza.

Definizione 2.1 (Test-Time Compute Budget)**.** Dato un modello π_θ e un problema x, il *test-time compute budget* C_{test} è il numero di token generati (o FLOPs) utilizzati prima di produrre la risposta finale. Il *test-time compute scaling law* è:

$$\text{Err}(C_{\text{test}}) \propto C_{\text{test}}^{-\alpha_{\text{test}}} \tag{2.1}$$

per qualche esponente $\alpha_{\text{test}} > 0$ dipendente dalla difficoltà del problema e dalla strategia di campionamento.

Utilità marginale del compute

Definiamo la *funzione di utilità marginale* del compute:

$$\frac{\partial\, \text{Acc}(C)}{\partial C} \tag{2.2}$$

dove $\text{Acc}(C)$ è l'accuratezza al compute C. Per il training compute:

$$\frac{\partial\, \text{Acc}(C_{\text{train}})}{\partial C_{\text{train}}} \propto C_{\text{train}}^{-\beta}, \quad \beta \approx 0.5\text{–}0.7, \tag{2.3}$$

(legge di potenza con esponente modesto). Per il test-time compute con best-of-N:

$$\frac{\partial\, \text{Acc}(N)}{\partial N} = p\,(1-p)^{N-1} \approx p\, e^{-Np}, \tag{2.4}$$

che decade esponenzialmente in N. Questo implica che per problemi con p piccolo (problemi difficili), il test-time compute è molto più efficiente del training compute nella fase iniziale, ma satura rapidamente.

§520 – Monte Carlo Tree Search per LLM

MCTS opera su un albero di ricerca $\mathcal{T}$ con nodi stati e archi azioni. Il ciclo standard ha quattro fasi:

Algorithm 1 MCTS per Generazione di Testo

Require: Modello π_θ, verifier V_ϕ, budget N_{sim}, problema x, costante $c > 0$

1: Inizializza la radice root $= (x, \varepsilon)$ con $n = 0, q = 0$
2: **for** $i = 1, \dots, N_{\text{sim}}$ **do**
3: **Selezione:** $v_l \leftarrow$ TreePolicy(root) usando UCB1: $a^* = \arg\max_a \left[q(a) + c\sqrt{\frac{\ln n_{\text{parent}}}{n(a)}} \right]$
4: **Espansione:** genera con π_θ un passo di ragionamento r come figlio di v_l
5: **Simulazione:** genera completamento fino a termine con π_θ
6: **Backpropagation:** $\delta \leftarrow V_\phi$(risposta finale); aggiorna $q(v)$ e $n(v)$ per tutti v sul percorso verso la radice
7: **end forreturn** $a^* = \arg\max_a n(\text{root}, a)$

UCB1 per token sequenziali

Nel contesto LLM, un "nodo" è un prefisso di ragionamento $p = (x, r_1, \dots, r_k)$. Il numero di figli possibili è $|V|$ (il vocabolario), quindi MCTS deve espandersi su un branching factor enorme. Nella pratica si campiona b figli dall'LLM invece di enumerarli:

$$\text{UCB1}(p, a) = \hat{q}(p, a) + c\sqrt{\frac{\ln n(p)}{n(p, a)}}, \tag{2.5}$$

dove $\hat{q}(p,a) = \frac{1}{n(p,a)} \sum_{i:\, \text{path} \ni (p,a)} \delta_i$ è la ricompensa media empirica.

Lemma 2.1 (Regret di UCB1)**.** Per UCB1 applicato a un problema a K bracci con orizzonte n, il regret atteso è:

$$\mathbb{E}\left[\sum_{t=1}^{n} (\mu^* - \mu_{a_t})\right] \le \sum_{i:\, \mu_i < \mu^*} \frac{8 \ln n}{\mu^* - \mu_i} + \left(1 + \frac{\pi^2}{3}\right) \sum_{i:\, \mu_i < \mu^*} (\mu^* - \mu_i), \tag{2.6}$$

ovvero $O(\ln n)$ contro $O(n)$ di una politica casuale.

Dimostrazione. Sia $\Delta_i = \mu^* - \mu_i$ il gap del braccio i. Il numero atteso di volte che il braccio subottimale i viene selezionato prima del tempo n è:

$$\mathbb{E}[T_i(n)] \le \frac{8 \ln n}{\Delta_i^2} + 1 + \frac{\pi^2}{3}.$$

Questo si dimostra bounding the probability that arm i is selected over the optimal arm using Hoeffding's inequality on the empirical means. Il regret totale è $\sum_i \Delta_i \cdot \mathbb{E}[T_i(n)]$, da cui il bound del lemma. $\square$

MCTS con politica di rollout guidata

Una variante importante è MCTS con *policy network*: invece di completare con π_θ standard (che potrebbe generare risposte di bassa qualità), si usa una politica di rollout π_{roll} specializzata:

$$\delta_{\text{hybrid}} = \lambda\, V_\phi(p) + (1-\lambda)\, R(\text{rollout}(\pi_{\text{roll}}, p)), \tag{2.7}$$

dove $\lambda \in [0,1]$ bilancia la stima del valore immediato e la ricompensa del rollout. Per $\lambda = 1$ si ottiene *value-only MCTS* (come in AlphaGo); per $\lambda = 0$ si ottiene *rollout-only MCTS*.

§521 – Process Reward Models

Definizione 2.2 (Process Reward Model)**.** Dato un problema x e una traiettoria di ragionamento $r = (r_1, \ldots, r_T)$, un PRM $V_\phi^{\text{PRM}} : \mathcal{X} \times \mathcal{R}^* \to [0,1]$ assegna $V_\phi^{\text{PRM}}(x, r_1, \ldots, r_t)$ come probabilità che il passo r_t sia corretto dato il contesto.

Addestramento dei PRM

Il PRM è addestrato su dataset di ragionamenti annotati passo per passo:

$$\mathcal{L}_{\text{PRM}}(\phi) = -\mathbb{E}_{(x,r,y)\sim\mathcal{D}} \sum_{t=1}^{T} \left[y_t \log V_\phi(x, r_{\le t}) + (1-y_t) \log(1 - V_\phi(x, r_{\le t})) \right], \tag{2.8}$$

dove $y_t \in \{0,1\}$ indica se il passo t è corretto.

Annotazione automatica tramite rollout

Per evitare l'annotazione umana passo per passo, si usa la tecnica dei *process labels da rollout*: $y_t = 1$ se esiste almeno un completamento da r_t che porta alla risposta corretta. Formalmente:

$$y_t = \mathbf{1}\left[\exists\, r_{t+1:T} \;:\; f(x, r_{1:T}) = y^*\right], \tag{2.9}$$

dove f è il verificatore deterministico e y^* è la risposta corretta. Questa annotazione è stimata via campionamento Monte Carlo: si generano M completamenti da r_t e si stima $\hat{y}_t = (1/M) \sum_m \mathbf{1}[f(x, r_{1:t}, r_{t+1:T}^{(m)}) = y^*]$.

Proposizione 2.1 (PRM come Stima del Vantaggio)**.** In un MDP a orizzonte T con ricompensa terminale $R \in \{0,1\}$:

$$V_\phi^{\text{PRM}}(x, r_{\le t}) \approx P(R = 1 \mid x, r_{\le t}) = V^*(b_t) \tag{2.10}$$

è la funzione valore ottimale per la belief state b_t condizionata sul prefisso osservato.

Confronto PRM vs ORM

Un *Outcome Reward Model* (ORM) assegna un singolo score alla risposta finale: $V_\phi^{\text{ORM}}(x, y) \in [0, 1]$. Il confronto formale è:

- **ORM**: segnale di addestramento più abbondante (ogni risposta), ma nessun segnale per i passi intermedi. Non può guidare MCTS al livello del passo.
- **PRM**: segnale sparso e costoso da annotare, ma fornisce feedback passo per passo. Permette MCTS a livello di ragionamento, con miglioramenti significativi su problemi matematici difficili (Lightman et al., 2023).

§522 – Best-of-N con Verifier: Analisi Statistica

La strategia più semplice di test-time compute: si campionano N completamenti indipendenti $y_1, \ldots, y_N \sim \pi_\theta(\cdot \mid x)$ e si restituisce quello con score massimo dal verifier:

$$y^* = \arg\max_{i=1,\ldots,N} V_\phi(x, y_i). \tag{2.11}$$

Analisi della performance

Sia $p = P(V_\phi(x, y) \geq \tau)$ la probabilità che un singolo campione superi la soglia τ. La probabilità che almeno un campione sia corretto è:

$$P(\text{almeno un corretto}) = 1 - (1 - p)^N. \tag{2.12}$$

Proposizione 2.2 (Scaling di Best-of-N)**.** Per p piccolo la probabilità di successo scala come:

$$P(\text{successo}) \approx 1 - e^{-Np}, \quad N \gg 1/p. \tag{2.13}$$

Per superare $1 - \delta$, servono $N^* = \lceil \ln(1/\delta)/p \rceil$ campioni.

Ottimizzazione del verifier

Un verifier imperfetto con precision $\rho = P(V_\phi = 1 \mid y = y^*)/P(V_\phi = 1)$ introduce un bias: best-of-N restituisce la risposta giusta solo se il verifier la identifica correttamente. La probabilità di successo effettiva è:

$$P(\text{successo effettivo}) = 1 - (1 - p\rho)^N. \tag{2.14}$$

Per $\rho < 1$, il verifier imperfetto degrada le prestazioni di un fattore ρ sull'esponente. Questo motiva il *verifier ensemble*: usare K verifier indipendenti con voto di maggioranza, ottenendo $\rho_{\text{ensemble}} \geq \rho$.

§523 – Beam Search come Caso Speciale di MCTS

La beam search standard con ampiezza B e lunghezza T è recuperabile come MCTS degenere con:

- profondità di espansione $= T$, branching factor $= B$;
- nessuna simulazione stocastica (greedy expansion);
- score dal log-likelihood del modello anziché da un verifier.

Il vantaggio di MCTS con PRM rispetto a beam search è formalmente catturabile: beam search massimizza $\prod_t \pi_\theta(r_t \mid \cdot)$ (la probabilità del prefisso), mentre MCTS con PRM massimizza $V_\phi(x, r_{\leq T})$ (la probabilità di correttezza della risposta finale). Le due quantità possono divergere significativamente: sequenze ad alta probabilità possono essere scorrette, e sequenze a bassa probabilità possono portare a risposte corrette.

Proposizione 2.3 (Gap tra Beam Search e MCTS)**.** Sia $G = \max_{r \in \mathcal{R}}[V_\phi(x, r) - \log \pi_\theta(r \mid x)]$ il massimo gap tra qualità e verosimiglianza. Beam search ottiene un gap atteso $\Omega(G)$ rispetto a MCTS quando $G > 0$.

§524 – Thinking Models: RL sul Ragionamento e GRPO

I modelli "thinking" (o1, Gemini Flash Thinking, QwQ) sono addestrati con RL a generare catene di pensiero lunghe prima della risposta finale.

Definizione 2.3 (Politica Thinking)**.** Sia x il problema, T la catena di pensiero (variabile) e y la risposta. La politica thinking fattorizza:

$$\pi_\theta(T, y \mid x) = \pi_\theta(T \mid x) \cdot \pi_\theta(y \mid x, T). \tag{2.15}$$

L'addestramento RL ottimizza:

$$J(\theta) = \mathbb{E}_{x \sim \mathcal{D},\, T \sim \pi_\theta(\cdot|x),\, y \sim \pi_\theta(\cdot|x,T)} \left[R(x, y) - \alpha \, \mathrm{KL}(\pi_\theta \| \pi_{\text{ref}}) \right]. \tag{2.16}$$

GRPO: Group Relative Policy Optimization

GRPO (DeepSeek, 2024) stima il vantaggio senza un critico separato, usando la media del gruppo:

$$\hat{A}_i = \frac{R_i - \text{mean}(R_1, \ldots, R_G)}{\text{std}(R_1, \ldots, R_G)}, \quad i = 1, \ldots, G, \tag{2.17}$$

dove G completamenti sono campionati per ogni problema. La loss è:

$$\mathcal{L}_{\text{GRPO}} = -\frac{1}{G} \sum_{i=1}^{G} \left[\min\Big(r_i \hat{A}_i,\ \text{clip}(r_i, 1-\varepsilon, 1+\varepsilon) \hat{A}_i \Big) \right] + \beta \, \mathrm{KL}(\pi_\theta \| \pi_{\text{ref}}), \tag{2.18}$$

con $r_i = \pi_\theta(T_i, y_i \mid x) / \pi_{\theta^{\text{old}}}(T_i, y_i \mid x)$.

Derivazione formale di GRPO dal principio di massimizzazione vincolata

GRPO si deriva come soluzione approssimata del problema:

$$\max_{\pi} \mathbb{E}_{\pi}[R(x, y)] \quad \text{s.t.} \quad \mathrm{KL}(\pi \| \pi_{\mathrm{ref}}) \leq \delta. \tag{2.19}$$

La soluzione esatta è $\pi^*(y \mid x) \propto \pi_{\mathrm{ref}}(y \mid x) \exp(R(x, y)/\beta)$ (distribuzione di Gibbs). GRPO approssima questa soluzione sostituendo $R(x, y)$ con il vantaggio normalizzato $\hat{A}_i$, che rimuove la varianza dell'estimatore riducendo il numero di campioni richiesti per la convergenza. La normalizzazione $\hat{A}_i$ corrisponde a centrare e ridimensionare le ricompense del gruppo, equivalente a una stima di baseline per il metodo REINFORCE.

Capitolo 3

Sezione C — Generazione Video e Modelli Spaziotemporali

3.1 Formalizzazione del Problema: Video come Sequenza 4D

§525 – Rappresentazione Spaziotemporale del Video

Un video è una funzione $v : \{1, \dots, T\} \times \{1, \dots, H\} \times \{1, \dots, W\} \to \mathbb{R}^3$ che mappa frame temporali × posizioni spaziali → valori RGB. La complessità computazionale è $O(T \cdot H \cdot W \cdot C)$: per un video di 10 secondi a 24fps in risoluzione 512×512, $10 \times 24 \times 512 \times 512 \times 3 \approx 188$ milioni di valori.

Definizione 3.1 (Video Tokenizer Spaziotemporale)**.** Un video tokenizer è una coppia (E_ϕ, D_ϕ) dove:

$$z = E_\phi(v) \in \mathbb{R}^{T_z \times H_z \times W_z \times d_z}, \qquad \hat{v} = D_\phi(z) \approx v, \tag{3.1}$$

con fattori di compressione $T/T_z = s_T$ temporale e $H/H_z = W/W_z = s_S$ spaziale. Tipicamente $s_T \in \{4, 8\}$ e $s_S \in \{8, 16\}$, riducendo il numero di token di un fattore $s_T \cdot s_S^2$.

La loss di addestramento del 3D-VAE è:

$$\mathcal{L}_{\text{3D-VAE}} = \underbrace{\mathbb{E}_v \left[\|v - \hat{v}\|_2^2 \right]}_{\text{ricostruzione}} + \beta \underbrace{\text{KL}(q_\phi(z \mid v) \| \mathcal{N}(0, I))}_{\text{regolarizzazione}} + \underbrace{\mathcal{L}_{\text{perc}} + \mathcal{L}_{\text{GAN}}}_{\text{qualità visiva}}. \tag{3.2}$$

3D-VAE: Encoder e Decoder

L'encoder E_ϕ è tipicamente una rete convoluzionale 3D con strati di *downsampling* temporale (stride s_T) e spaziale (stride s_S):

$$E_\phi = \text{Conv3D}_{s_T, s_S, s_S} \circ \text{ResBlock3D} \circ \cdots \tag{3.3}$$

Il decoder D_ϕ usa *upsampling* bilineare 3D seguito da convoluzione:

$$D_\phi = \text{Conv3D} \circ \text{Upsample}_{s_T, s_S, s_S} \circ \text{ResBlock3D} \circ \cdots \tag{3.4}$$

La loss percettuale $\mathcal{L}_{\text{perc}}$ calcola la distanza tra feature estratte da una rete VGG preaddestrata, frame per frame:

$$\mathcal{L}_{\text{perc}} = \mathbb{E}_v \left[\frac{1}{T} \sum_{t=1}^{T} \|\phi_{\text{VGG}}(v_t) - \phi_{\text{VGG}}(\hat{v}_t)\|_1 \right]. \tag{3.5}$$

§526 – Video Diffusion Models: Derivazione Completa

Il Video Diffusion Model (Ho et al., 2022) estende DDPM (Vol. III, §101–108) ai video trattando ogni video come un'unità da denoisare congiuntamente.

Processo di forward diffusion per video

Il processo di diffusione per un video v_0 aggiunge rumore gaussiano per K passi:

$$q(v_{1:K} \mid v_0) = \prod_{k=1}^{K} q(v_k \mid v_{k-1}), \qquad q(v_k \mid v_{k-1}) = \mathcal{N}(v_k;\, \sqrt{1-\beta_k}\, v_{k-1},\, \beta_k I). \tag{3.6}$$

La distribuzione marginale ha forma chiusa:

$$q(v_k \mid v_0) = \mathcal{N}(v_k;\, \sqrt{\bar{\alpha}_k}\, v_0,\, (1-\bar{\alpha}_k) I), \qquad \bar{\alpha}_k = \prod_{j=1}^{k} (1-\beta_j). \tag{3.7}$$

Derivazione della loss ELBO

La lower bound sull'evidence si deriva come per DDPM, ma ora v_0 è un tensore $T \times H \times W \times 3$:

$$\begin{aligned} \log p_\theta(v_0) &\geq \mathbb{E}_q \left[\log \frac{p_\theta(v_{0:K})}{q(v_{1:K} \mid v_0)} \right] \\ &= -\sum_{k=2}^{K} D_{\mathrm{KL}}(q(v_{k-1} \mid v_k, v_0) \| p_\theta(v_{k-1} \mid v_k)) + \mathbb{E}_q[\log p_\theta(v_0 \mid v_1)]. \end{aligned} \tag{3.8}$$

Parametrizzando il denoiser come predizione del rumore $\epsilon_\theta(v_k, k, c)$:

$$\mathcal{L}_{\mathrm{VDM}} = \mathbb{E}_{k, v_0, \epsilon} \left[\left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_k}\, v_0 + \sqrt{1-\bar{\alpha}_k}\, \epsilon,\, k,\, c) \right\|_2^2 \right]. \tag{3.9}$$

Architettura del denoiser per video

Definizione 3.2 (Attenzione Spaziotemporale Fattorizzata)**.** L'attenzione congiunta su $T \times H \times W$ token ha costo $O((THW)^2)$. L'attenzione fattorizzata suddivide il calcolo:

$$Z^{\mathrm{spatial}} = \mathrm{SpatialAttn}(V_k), \qquad \text{costo: } O(T \cdot (HW)^2), \tag{3.10}$$

$$Z^{\mathrm{temporal}} = \mathrm{TemporalAttn}(Z^{\mathrm{spatial}}), \qquad \text{costo: } O(HW \cdot T^2), \tag{3.11}$$

con costo totale $O(T(HW)^2 + HW\,T^2) \ll O((THW)^2)$ per $T \ll HW$.

Proposizione 3.1 (Risparmio computazionale)**.** Per $T = 16$ frame e $H = W = 64$ pixel latenti ($HW = 4096$):

$$\begin{aligned} \text{Attenzione congiunta:} &\quad (16 \cdot 4096)^2 = 4.3 \times 10^9 \text{ operazioni,} \\ \text{Attenzione fattorizzata:} &\quad 16 \cdot 4096^2 + 4096 \cdot 256 = 2.7 \times 10^8 \text{ operazioni,} \end{aligned}$$

ovvero un risparmio di circa 16×.

§527 – Sora e Diffusion Transformer Spaziotemporale

Sora (OpenAI, 2024) utilizza un Diffusion Transformer (DiT) spaziotemporale operante su patch video compressi.

Patchification 3D

Un video latente $z \in \mathbb{R}^{T_z \times H_z \times W_z \times d_z}$ viene diviso in patch 3D di dimensione $p_t \times p_h \times p_w$, producendo:

$$N_{\text{patch}} = \frac{T_z}{p_t} \cdot \frac{H_z}{p_h} \cdot \frac{W_z}{p_w} \quad \text{token, ciascuno proiettato a dimensione } d. \tag{3.12}$$

Positional encoding spaziotemporale

Ogni patch (t_i, h_i, w_i) riceve un encoding posizionale additivo:

$$\text{PE}(t, h, w) = \text{PE}_t(t) + \text{PE}_h(h) + \text{PE}_w(w), \tag{3.13}$$

Con RoPE 3D, l'encoding è codificato nelle fasi della rotazione del prodotto scalare, estendendo la formulazione di RoPE (Vol. VIII, §448.2) a tre dimensioni. Per il vettore query q e il vettore key k in posizione (t, h, w):

$$\text{Attn}(q, k) = q^\top R_t(t)\, R_h(h)\, R_w(w)\, k, \tag{3.14}$$

dove R_t, R_h, R_w sono matrici di rotazione in sottospazi ortogonali di $\mathbb{R}^d$.

Condizionamento testuale con adaLN

Il condizionamento sul prompt avviene tramite adaptive layer norm:

$$\text{adaLN}(h, c) = \gamma(c) \odot \frac{h - \mu}{\sigma} + \beta(c), \tag{3.15}$$

dove $\gamma(c), \beta(c) \in \mathbb{R}^d$ sono prodotti da un MLP applicato all'embedding del condizionamento c (embedding CLIP del testo concatenato all'embedding del passo di diffusione k). L'adaLN permette al modello di scalare e traslare le attivazioni in modo input-dipendente, ottenendo un condizionamento più potente rispetto all'iniezione tramite cross-attention.

Flow Matching per Video

Un'alternativa al paradigma DDPM è il *flow matching* (Lipman et al., 2022), che apprende un campo di velocità $u_\theta(v, t)$ per trasformare il rumore in video:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t\sim\mathcal{U}[0,1],\, v_0\sim p_{\text{data}},\, \epsilon\sim\mathcal{N}(0,I)} \left[\| u_\theta(v_t, t) - (v_0 - \epsilon) \|_2^2 \right], \tag{3.16}$$

dove $v_t = (1-t)\epsilon + t\, v_0$ è l'interpolazione lineare. Il vantaggio rispetto a DDPM è la traiettoria rettilinea (invece di curvilinea), che permette generazione di alta qualità in pochi passi di integrazione.

§528 – VideoGPT: Generazione Autoregressiva su Token Video

VideoGPT (Yan et al., 2021) combina VQ-VAE 3D (Vol. III, §101) con un trasformatore autoregressivo.

VQ-VAE 3D per video

Il quantizzatore vettoriale mappa ogni latente z_i al token più vicino nel codebook $\mathcal{C} = \{e_1, \ldots, e_K\} \subset \mathbb{R}^{d_z}$:

$$\hat{z}_i = e_{k^*}, \qquad k^* = \arg\min_k \| z_i - e_k \|_2, \tag{3.17}$$

$$\mathcal{L}_{\text{VQ}} = \| v - D_\phi(\hat{z}) \|_2^2 + \| \text{sg}[z] - \hat{z} \|_2^2 + \beta \, \| z - \text{sg}[\hat{z}] \|_2^2, \tag{3.18}$$

dove $\text{sg}[\cdot]$ è lo stop-gradient. Il secondo termine aggiorna il codebook, il terzo (commitment loss) incentiva l'encoder a rimanere vicino al codebook.

GPT autoregressivo sullo spazio discreto

Dopo la VQ-VAE, un video è rappresentato come sequenza $\mathbf{q} = (q_1, \ldots, q_{N_{\text{patch}}})$ con $q_i \in \{1, \ldots, K\}$. Un GPT causale (Vol. II, §56) modella:

$$P(\mathbf{q}) = \prod_{i=1}^{N_{\text{patch}}} P(q_i \mid q_1, \ldots, q_{i-1}), \qquad \mathcal{L}_{\text{GPT}} = -\sum_{i=1}^{N_{\text{patch}}} \log P(q_i \mid q_{<i}). \tag{3.19}$$

§529 – Consistency Models per la Generazione Video

Definizione 3.3 (Consistency Function per Video)**.** Una funzione di consistenza video è una mappa $f_\theta(v_k, k) : \mathbb{R}^{THW} \times \mathbb{R} \to \mathbb{R}^{THW}$ tale che:

$$f_\theta(v_k, k) = f_\theta(v_{k'}, k') \quad \forall\, k, k' \text{ lungo la stessa traiettoria ODE.} \tag{3.20}$$

La loss di consistency per video è:

$$\mathcal{L}_{\text{CM-video}} = \mathbb{E}_{k, v_0, \epsilon} \left[d(f_\theta(v_k, k),\ \text{sg}[f_{\theta^-}(\hat{v}_{k-1}, k-1)]) \right], \tag{3.21}$$

dove $\hat{v}_{k-1} = v_k - (k - k - 1)\, \epsilon_\phi(v_k, k)$ è un passo della ODE di diffusione e θ^- sono i parametri EMA. La funzione distanza $d(\cdot, \cdot)$ è tipicamente LPIPS (Learned Perceptual Image Patch Similarity) per catturare la qualità percettiva piuttosto che la distanza ℓ_2 pixel per pixel.

Confronto computazionale tra architetture

Tabella 3.1: Confronto tra le architetture di generazione video trattate.

Modello	**Passi inf.**	**Token tipo**	**Cond.**	**Note**
VDM	50–1000	continui	testo	DDPM latente
Sora/DiT	20–50	patch 3D	testo	RoPE 3D, adaLN
VideoGPT	1 (AR)	discreti	classe	VQ-VAE + GPT
CM-video	1–4	continui	testo	distillazione ODE

Capitolo 4

Sezione D — Continual e Lifelong Learning

4.1 Catastrophic Forgetting: Analisi Teorica

§530 – Il Problema del Forgetting: Definizione Formale

Definizione 4.1 (Catastrophic Forgetting)**.** Siano $\mathcal{T}_1, \mathcal{T}_2, \ldots$ una sequenza di task con distribuzioni $\mathcal{D}_1, \mathcal{D}_2, \ldots$. Un modello f_θ è soggetto a *catastrophic forgetting* se, dopo l'addestramento sequenziale su $\mathcal{T}_1, \ldots, \mathcal{T}_t$:

$$\mathcal{L}_i(f_\theta) \gg \mathcal{L}_i(f_{\theta_i^*}) \quad \exists\, i < t, \tag{4.1}$$

dove θ_i^* sono i parametri ottimali per il solo task i. Il *forgetting score* al passo t è:

$$\text{FGT}_t = \frac{1}{t-1} \sum_{i=1}^{t-1} \left[\mathcal{L}_i(f_\theta^{(t)}) - \mathcal{L}_i(f_\theta^{(i)}) \right]. \tag{4.2}$$

Analisi geometrica del forgetting

Il catastrophic forgetting nasce dalla geometria dello spazio dei parametri. Sia $\mathcal{L}_1(\theta)$ la loss del task 1 e $\mathcal{L}_2(\theta)$ quella del task 2. L'approssimazione di Taylor al second'ordine di $\mathcal{L}_1$ intorno a θ_1^*:

$$\mathcal{L}_1(\theta_2^*) \approx \mathcal{L}_1(\theta_1^*) + \frac{1}{2}(\theta_2^* - \theta_1^*)^\top F_1 (\theta_2^* - \theta_1^*), \tag{4.3}$$

dove $F_1 = \nabla^2_\theta \mathcal{L}_1|_{\theta=\theta_1^*}$ è la matrice hessiana del task 1. Il forgetting è grave quando $\theta_2^* - \theta_1^*$ ha componenti grandi lungo le direzioni con grande curvatura di $\mathcal{L}_1$.

Tassonomy del forgetting

- **Interferenza catastrofica**: $\theta_2^* \approx -\theta_1^*$ (minimi opposti). Causa forgetting totale.
- **Interferenza parziale**: $\theta_2^* - \theta_1^*$ è ortogonale ad alcune direzioni di alta curvatura. Causa forgetting selettivo.
- **Transfer positivo**: θ_2^* giace in un bacino che include anche θ_1^*. Causa miglioramento retroattivo (*backward transfer*).

§531 – Teoria PAC su Distribuzioni Non Stazionarie

Il framework PAC (Vol. VI, §347) assume i.i.d. da una distribuzione fissa $\mathcal{D}$. Nel continual learning la distribuzione cambia nel tempo.

Definizione 4.2 (Non-Stazionarietà e Concept Drift)**.** Una sequenza di distribuzioni $(\mathcal{D}_t)_{t\geq 1}$ presenta *concept drift* se esiste t^* tale che $\mathcal{D}_{t^*+1} \neq \mathcal{D}_{t^*}$. Il drift è *brusco* se $\mathrm{TV}(\mathcal{D}_t, \mathcal{D}_{t+1}) > \delta$ per un solo t^*, *graduale* se la variazione è distribuita nel tempo.

Teorema 4.1 (Complessità di Campionamento con Concept Drift)**.** Per un apprendimento online su un flusso di dati con drift brusco al passo t^*, affinché l'errore dopo il drift scenda sotto ε con probabilità $1-\delta$, servono:

$$m^* = O\left(\frac{d_{\mathrm{VC}} \log(1/\varepsilon) + \log(1/\delta)}{\varepsilon}\right) \tag{4.4}$$

campioni dopo il drift, indipendentemente dalla quantità di dati osservati prima.

Dimostrazione. [Sketch] Dopo il drift, la distribuzione è $\mathcal{D}_{t^*+1}$ completamente nuova. Il teorema PAC standard (Vol. VI, §347) si applica direttamente con m^* campioni i.i.d. da $\mathcal{D}_{t^*+1}$. I campioni precedenti sono irrilevanti poiché provengono da $\mathcal{D}_{t^*}$ con $\mathrm{TV}(\mathcal{D}_{t^*}, \mathcal{D}_{t^*+1}) > \delta$: il loro contributo al rischio empirico su $\mathcal{D}_{t^*+1}$ può peggiorare la stima. □

Drift graduale e finestre scorrevoli

Per il drift graduale, la soluzione è usare una *finestra scorrevole* di dimensione W: si usa solo i dati raccolti negli ultimi W passi. Il trade-off è:

- W grande: bias basso (più dati), ma varianza alta rispetto al drift ($\mathcal{D}_{t-W} \neq \mathcal{D}_t$ se il drift è rapido).
- W piccolo: bias alto (meno dati), ma risposta rapida al drift.

Il W ottimale per un drift graduale con tasso r_{drift} è $W^* = O((d_{\mathrm{VC}}/r_{\mathrm{drift}})^{1/2})$.

§532 – Elastic Weight Consolidation: Derivazione Bayesiana

L'Elastic Weight Consolidation (EWC; Kirkpatrick et al., 2017) è un metodo di regolarizzazione che preserva i parametri importanti per i task precedenti.

Derivazione MAP sequenziale

Applicando la regola di Bayes in modo sequenziale:

$$P(\theta \mid \mathcal{D}_1, \mathcal{D}_2) \propto P(\mathcal{D}_2 \mid \theta) \cdot P(\theta \mid \mathcal{D}_1). \tag{4.5}$$

L'ingrediente chiave è approssimare il posteriore $P(\theta \mid \mathcal{D}_1)$ con una gaussiana centrata sull'ottimo θ_1^*:

$$P(\theta \mid \mathcal{D}_1) \approx \mathcal{N}(\theta;\, \theta_1^*,\, F_1^{-1}), \tag{4.6}$$

dove F_1 è la matrice di informazione di Fisher del task 1 (Vol. IV, §202).

Definizione 4.3 (Loss EWC)**.** La loss EWC per il task 2 è:

$$\mathcal{L}_{\text{EWC}}(\theta) = \mathcal{L}_2(\theta) + \frac{\lambda}{2} \sum_j F_{1,jj}\,(\theta_j - \theta_{1,j}^*)^2, \tag{4.7}$$

dove $F_{1,jj}$ è il j-esimo elemento diagonale della matrice di Fisher di task 1.

Calcolo della matrice di Fisher diagonale

La matrice di Fisher completa $F_1 \in \mathbb{R}^{P \times P}$ è:

$$F_1 = \mathbb{E}_{x \sim \mathcal{D}_1} \left[\nabla_\theta \log P(y \mid x, \theta_1^*)\, \nabla_\theta \log P(y \mid x, \theta_1^*)^\top \right]. \tag{4.8}$$

Si usa l'approssimazione diagonale per scalabilità:

$$F_{1,jj} = \mathbb{E}_{x \sim \mathcal{D}_1} \left[\left(\frac{\partial \log P(y \mid x, \theta_1^*)}{\partial \theta_j} \right)^2 \right]. \tag{4.9}$$

Il costo è $O(P)$ in memoria (anziché $O(P^2)$ per la Fisher completa) e $O(|\mathcal{D}_1'| \cdot P)$ in tempo.

Proposizione 4.1 (EWC come Regolarizzazione Quadratica Ottimale)**.** Sotto l'approssimazione gaussiana $P(\theta \mid \mathcal{D}_1) \approx \mathcal{N}(\theta_1^*, F_1^{-1})$, la stima MAP del posteriore congiunto $P(\theta \mid \mathcal{D}_1, \mathcal{D}_2)$ è equivalente alla minimizzazione di (4.7) con $\lambda = 1$ e Fisher completa. L'approssimazione diagonale introduce un errore $O\Big(\sum_{i \neq j} F_{1,ij}(\theta_i - \theta_{1,i}^*)(\theta_j - \theta_{1,j}^*)\Big)$.

Online EWC e Synaptic Intelligence

Online EWC (Schwarz et al., 2018) mantiene una singola Fisher media su tutti i task precedenti invece di una per task:

$$\tilde{F} \leftarrow (1 - \alpha)\tilde{F} + \alpha\, F_t, \qquad \alpha \in (0, 1), \tag{4.10}$$

riducendo il costo da $O(T \cdot P)$ a $O(P)$ in memoria.

Synaptic Intelligence (SI; Zenke et al., 2017) stima online l'importanza dei parametri durante l'addestramento, senza calcolare esplicitamente la Fisher:

$$\omega_j^{(k)} = \frac{\sum_{t:\,\text{task } k} \delta_{j,t} \cdot \Delta\theta_{j,t}}{(\theta_j^{(k)} - \theta_j^{(k-1)})^2 + \xi}, \tag{4.11}$$

dove $\delta_{j,t} = -\frac{\partial \mathcal{L}}{\partial \theta_j}\Big|_t$ è il contributo del parametro j alla riduzione della loss al passo t, e $\xi > 0$ è un termine di regolarizzazione. La loss SI è analoga a EWC con $F_{1,jj}$ sostituito da ω_j.

§533 – Progressive Neural Networks

Le Progressive Neural Networks (PNN; Rusu et al., 2016) evitano il forgetting in modo radicale: invece di modificare i pesi precedenti, si aggiunge un'intera nuova colonna di neuroni per ogni nuovo task.

Definizione 4.4 (Colonna Progressiva)**.** Dopo K task, l'architettura PNN ha K colonne $\{f_{\theta^{(k)}}\}_{k=1}^{K}$. L'output dello strato l del task k è:

$$h^{(k,l)} = \sigma\left(W^{(k,l)} h^{(k,l-1)} + \sum_{j<k} U^{(k,j,l)} h^{(j,l-1)} \right), \tag{4.12}$$

dove $W^{(k,l)}$ sono i nuovi pesi della colonna k e $U^{(k,j,l)}$ sono i pesi delle *lateral connections* dalla colonna j alla colonna k. I pesi $W^{(j,\cdot)}$ con $j < k$ sono congelati.

Analisi del transfer positivo

Il gradiente rispetto ai pesi della colonna k è:

$$\frac{\partial \mathcal{L}_k}{\partial W^{(k,l)}} = \frac{\partial \mathcal{L}_k}{\partial h^{(k,l)}} \cdot \left(h^{(k,l-1)}\right)^\top, \tag{4.13}$$

che non dipende dalle colonne precedenti (il gradiente non fluisce verso i pesi congelati). Il costo in memoria cresce linearmente: $O(K \cdot P_{\text{colonna}})$.

PackNet: pruning e packing

PackNet (Mallya e Lazebnik, 2018) è un'alternativa alle PNN che evita la crescita lineare della memoria tramite *pruning iterativo*: per ogni nuovo task si identificano i pesi meno importanti per i task precedenti (usando magnitude pruning) e si usano per il nuovo task:

1. Addestra su task k usando tutti i pesi.
2. Applica pruning: congela i pesi che vanno nel top-$\rho\%$ dei pesi importanti per task $< k$ (secondo una metrica di importanza come $|w_j| / \max_j |w_j|$).
3. Ri-addestra solo i pesi non congelati per task k.

Il costo in memoria è $O(P)$ (rete fissa), ma la capacità per task decresce con il numero di task come P/K.

§534 – Gradient Episodic Memory (GEM) e A-GEM

Definizione 4.5 (GEM (Lopez-Paz e Ranzato, 2017))**.** Si mantiene un buffer di memoria $\mathcal{M}_k$ per ogni task $k < t$. Al passo di aggiornamento su task t, si risolve il problema vincolato:

$$\tilde{g} = \arg\min_{g} \|g - g_t\|_2^2 \quad \text{s.t.} \quad \langle g,\, g_k \rangle \geq 0 \quad \forall\, k < t, \tag{4.14}$$

dove $g_t = \nabla_\theta \mathcal{L}_t$ e $g_k = \nabla_\theta \mathcal{L}(\theta; \mathcal{M}_k)$.

Teorema 4.2 (GEM garantisce non-worsening)**.** Sia $\tilde{g}$ la soluzione di (4.14). Allora:

$$\mathcal{L}(\theta_{t+1}; \mathcal{M}_k) \leq \mathcal{L}(\theta_t; \mathcal{M}_k) + O(\eta^2) \quad \forall\, k < t. \tag{4.15}$$

Dimostrazione. Con aggiornamento $\theta_{t+1} = \theta_t - \eta\tilde{g}$:

$$\mathcal{L}(\theta_{t+1}; \mathcal{M}_k) \approx \mathcal{L}(\theta_t; \mathcal{M}_k) - \eta\langle g_k,\, \tilde{g} \rangle + O(\eta^2).$$

Il vincolo $\langle g_k,\, \tilde{g} \rangle \geq 0$ garantisce che il termine lineare sia non positivo, ovvero che la loss sul task k non aumenti al prim'ordine. □

Soluzione del QP in GEM

Il problema (4.14) è un QP convesso con $t-1$ vincoli lineari. La soluzione si trova tramite proiezione duale: la soluzione primale è $\tilde{g} = g_t + C\lambda^*$, dove $C = [g_1, \ldots, g_{t-1}]$ è la matrice dei gradienti di memoria e $\lambda^* \geq 0$ è il moltiplicatore di Lagrange ottimale. Il costo è $O((t-1) \cdot P)$ per il prodotto matrice-vettore.

A-GEM: Averaged GEM

A-GEM (Chaudhry et al., 2019) sostituisce i $t-1$ vincoli di GEM con un singolo vincolo medio, riducendo drasticamente il costo computazionale:

$$\tilde{g} = \begin{cases} g_t & \text{se } \langle g_t,\, g_{\text{ref}} \rangle \geq 0, \\ g_t - \dfrac{\langle g_t,\, g_{\text{ref}} \rangle}{\langle g_{\text{ref}},\, g_{\text{ref}} \rangle} g_{\text{ref}} & \text{altrimenti,} \end{cases} \tag{4.16}$$

dove $g_{\text{ref}} = \frac{1}{t-1} \sum_{k<t} g_k$ è il gradiente medio sulla memoria. Il costo scende a $O(P)$ (proiezione su un singolo iperpiano). A-GEM perde la garanzia per-task di GEM, ma mantiene il non-peggioramento in media:

$$\frac{1}{t-1} \sum_{k<t} \mathcal{L}(\theta_{t+1}; \mathcal{M}_k) \leq \frac{1}{t-1} \sum_{k<t} \mathcal{L}(\theta_t; \mathcal{M}_k) + O(\eta^2). \tag{4.17}$$

Capitolo 5

Sezione E — Topologia Applicata al Deep Learning

5.1 Fondamenti di Omologia Simpliciale

§535 – Complessi Simpliciali: Definizioni Fondamentali

La Topological Data Analysis (TDA) studia la forma globale dei dati usando strumenti dell'algebra topologica. Il concetto base è il complesso simpliciale, generalizzazione di un grafo.

Definizione 5.1 (Simplesso e Complesso Simpliciale)**.** Un *k-simplesso* è l'inviluppo convesso di $k+1$ punti geometricamente indipendenti $v_0, \dots, v_k \in \mathbb{R}^n$, denotato $\sigma = [v_0, \dots, v_k]$. I suoi *facce* sono i sub-simplessi $[v_{i_0}, \dots, v_{i_j}]$ con $j < k$.

Un *complesso simpliciale* K è una collezione di simplessi tale che:

1. Se $\sigma \in K$ e τ è una faccia di σ, allora $\tau \in K$;
2. Se $\sigma, \tau \in K$, allora $\sigma \cap \tau$ è una faccia comune o $\emptyset$.

Euler characteristic

Per un complesso simpliciale finito, la *caratteristica di Eulero* è:

$$\chi(K) = \sum_{k=0}^{\dim K} (-1)^k \, |C_k(K)| = \sum_{k=0}^{\dim K} (-1)^k \, \beta_k, \tag{5.1}$$

dove la seconda uguaglianza è il *Teorema di Eulero-Poincaré*. Per un triangolo ($K =$ un 2-simplesso e tutte le sue facce): $\chi = 1 - 3 + 3 - 1 = 0$? No: $\chi = 3 - 3 + 1 = 1$ (3 vertici, 3 spigoli, 1 triangolo) — consistente con $\beta_0 = 1, \beta_1 = 0$.

§536 – Omologia: Cicli, Bordi e Spazi di Betti

Definizione 5.2 (Catena, Ciclo e Bordo)**.** Il gruppo delle *k-catene* $C_k(K; \mathbb{F})$ è lo spazio vettoriale su un campo $\mathbb{F}$ con base i k-simplessi di K. L'operatore bordo $\partial_k : C_k \to C_{k-1}$ è:

$$\partial_k([v_0, \dots, v_k]) = \sum_{i=0}^{k} (-1)^i [v_0, \dots, \hat{v}_i, \dots, v_k], \tag{5.2}$$

dove $\hat{v}_i$ denota l'omissione di v_i.

Lemma 5.1 (Fondamentale: $\partial_{k-1} \circ \partial_k = 0$)**.** Per ogni k, il bordo del bordo è zero.

Dimostrazione. Calcoliamo direttamente:

$$\begin{aligned}\partial_{k-1}\partial_k[v_0,\dots,v_k] &= \sum_{i=0}^{k}(-1)^i \partial_{k-1}[v_0,\dots,\hat{v}_i,\dots,v_k] \\ &= \sum_{i=0}^{k}\sum_{j<i}(-1)^i(-1)^j[v_0,\dots,\hat{v}_j,\dots,\hat{v}_i,\dots,v_k] \\ &\quad + \sum_{i=0}^{k}\sum_{j>i}(-1)^i(-1)^{j-1}[v_0,\dots,\hat{v}_i,\dots,\hat{v}_j,\dots,v_k].\end{aligned}$$

Ogni coppia (i,j) con $i<j$ produce i termini $(-1)^{i+j}$ e $(-1)^{i+j-1}\cdot(-1) = (-1)^{i+j}$... a segno opposto, poiché il secondo termine (con $j'=i, i'=j$) ha segno $(-1)^{j+(i-1)} = (-1)^{i+j-1}$. I due contributi si cancellano a due a due, e la somma totale è zero. □

Definizione 5.3 (Gruppi di Omologia e Numeri di Betti)**.** Il k-esimo gruppo di omologia è:

$$H_k(K;\mathbb{F}) = \ker(\partial_k)/\operatorname{im}(\partial_{k+1}). \tag{5.3}$$

Il k-esimo numero di Betti è $\beta_k = \dim H_k(K;\mathbb{F})$. Interpretazione: β_0 = componenti connesse, β_1 = cicli indipendenti, β_2 = cavità chiuse.

Esempio 5.1 (Cerchio S^1)**.** Il cerchio triangolato ha $\beta_0 = 1$ (una componente connessa) e $\beta_1 = 1$ (un ciclo). Formalmente, con vertici $\{a,b,c\}$ e spigoli $\{ab, bc, ca\}$:

- $\ker \partial_1 = \operatorname{span}\{ab+bc+ca\}$ (l'unico ciclo)
- $\operatorname{im}\partial_2 = \{0\}$ (non ci sono 2-simplessi)
- $H_1 = \ker\partial_1/\operatorname{im}\partial_2 \cong \mathbb{F}$, quindi $\beta_1 = 1$.

§537 – Omologia Persistente: il Teorema di Struttura

Definizione 5.4 (Complesso di Vietoris-Rips)**.** Per una nuvola di punti $\mathcal{X}$ e un parametro $\varepsilon \geq 0$:

$$\mathrm{VR}(\mathcal{X},\varepsilon) = \{\sigma \subseteq \mathcal{X} : d(x_i,x_j) \leq \varepsilon \ \forall\, x_i,x_j \in \sigma\}. \tag{5.4}$$

Al crescere di ε si ottiene una filtrazione: $\mathrm{VR}(\mathcal{X},0) \subseteq \mathrm{VR}(\mathcal{X},\varepsilon_1) \subseteq \cdots$.

Definizione 5.5 (Diagramma di Persistenza)**.** Una feature topologica k-dimensionale nasce a ε_b e muore a $\varepsilon_d > \varepsilon_b$. Il diagramma di persistenza $\mathrm{Dgm}_k(\mathcal{X})$ è l'insieme dei punti $(b, d) \in \mathbb{R}^2$ con $b < d$, uno per ogni feature di dimensione k. La *persistenza* è $d - b$.

Teorema 5.1 (Struttura della Persistent Homology (Zomorodian e Carlsson, 2005))**.** Sia $K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m$ una filtrazione di complessi su un campo $\mathbb{F}$. I moduli di persistenza $\{H_k(K_i; \mathbb{F})\}$ si decompongono univocamente:

$$H_k\left(\bigcup_i K_i;\, \mathbb{F}\right) \cong \bigoplus_j \mathbb{F}[b_j, d_j), \tag{5.5}$$

dove $\mathbb{F}[b, d)$ è il modulo che è $\mathbb{F}$ per $b \leq i < d$ e 0 altrove.

Persistenza vettoriale

Per usare i diagrammi di persistenza come input per algoritmi di machine learning, si trasformano in vettori. Due tecniche comuni:

Persistence image (Adams et al., 2017): si costruisce un'immagine discreta della probabilità di persistenza tramite un kernel gaussiano centrato su ogni punto (b, d):

$$I(z) = \sum_{(b,d)\in \mathrm{Dgm}} w(b, d)\, \mathcal{N}(z;\, (b, d),\, \sigma^2 I_2), \quad w(b, d) = (d - b)^p, \tag{5.6}$$

dove $p > 0$ è un peso che valorizza le feature più persistenti.

Persistence landscape (Bubenik, 2015): la landscape $\lambda_k(t)$ è la k-esima funzione più alta tra le "tende" costruite sui punti del diagramma:

$$\lambda_k(t) = \mathrm{kmax}_{(b,d)\in \mathrm{Dgm}} \min(t - b, d - t)_+, \tag{5.7}$$

che è un elemento di L^2, quindi ammette media, varianza e test di permutazione.

§538 – Stabilità del Diagramma di Persistenza

Definizione 5.6 (Distanza Bottleneck)**.** Dati due diagrammi A e B, la distanza bottleneck è:

$$d_\infty(A, B) = \inf_{\eta: A \to B} \sup_{p \in A} \|p - \eta(p)\|_\infty\,, \tag{5.8}$$

dove l'inf è su tutte le biiezioni η (che possono mappare punti alla diagonale).

Teorema 5.2 (Stabilità (Cohen-Steiner et al., 2007))**.** Siano $f, g : K \to \mathbb{R}$ funzioni di Morse su un complesso simpliciale. Allora:

$$d_\infty(\mathrm{Dgm}(f), \mathrm{Dgm}(g)) \leq \|f - g\|_\infty . \tag{5.9}$$

Il teorema di stabilità garantisce che piccole perturbazioni dell'input producono piccole variazioni nel diagramma di persistenza, rendendo la TDA robusta al rumore.

Distanza Wasserstein tra diagrammi

Una metrica alternativa alla bottleneck è la *distanza di Wasserstein di ordine* q:

$$W_q(A, B) = \left(\inf_{\eta : A \to B} \sum_{p \in A} \|p - \eta(p)\|_\infty^q \right)^{1/q} . \tag{5.10}$$

Per $q \to \infty$ si recupera la distanza bottleneck. La W_2 è più sensibile alle differenze globali tra diagrammi e ammette un teorema di stabilità analogo: $W_2(\mathrm{Dgm}(f), \mathrm{Dgm}(g)) \leq C \|f - g\|_2$.

§539 – Numeri di Betti dei Loss Landscape

Definizione 5.7 (Sottoinsieme di Livello della Loss)**.** Per una loss $\mathcal{L} : \mathbb{R}^P \to \mathbb{R}$ e una soglia $c \in \mathbb{R}$:

$$\mathcal{L}^{-1}(-\infty, c] = \{\theta \in \mathbb{R}^P : \mathcal{L}(\theta) \leq c\}. \tag{5.11}$$

Risultati empirici (Nguyen e Hein, 2018; Gotmare et al., 2018) mostrano:

- Per reti sufficientemente overparametrizzate, $\beta_0 = 1$ per c poco superiore al minimo globale: tutti i minimi sono connessi.
- β_1 (buchi 1D) è elevato nella fase iniziale e decresce durante la convergenza: il cammino di discesa del gradiente "riempie" i buchi.

Connessione con il Teorema di Morse

Per una funzione di Morse $\mathcal{L}$ (con Hessiana non degenere in ogni punto critico), il k-esimo numero di Betti soddisfa:

$$\beta_k \leq C_k(\mathcal{L}), \tag{5.12}$$

dove $C_k(\mathcal{L})$ è il numero di punti critici di indice k (cioè con k autovalori negativi dell'Hessiana). I punti di sella con indice 1 contribuiscono a β_0 e β_1 nei sottoinsiemi di livello.

Fasi topologiche dell'addestramento

La topologia del loss landscape cambia durante l'addestramento in fasi distinte:

1. **Fase di inizializzazione**: $\beta_0 \gg 1$ (molti minimi locali disconnessi). Il paesaggio è altamente non convesso.
2. **Fase di discesa rapida**: i minimi locali si connettono man mano che c scende; β_0 decresce verso 1.
3. **Fase di convergenza**: $\beta_0 = 1$, β_1 basso. Il paesaggio è quasi convesso nell'intorno del minimo raggiunto.
4. **Fase di overfitting**: β_0 rimane 1 ma la loss di test aumenta, indicando che il modello ha trovato un minimo stretto (alta curvatura).

§540 – Il Mapper Algorithm: Visualizzazione Topologica

Algorithm 2 Mapper Algorithm

Require: Nuvola di punti $\mathcal{X}$, filtro $f : \mathcal{X} \to \mathbb{R}^d$, copertura $\mathcal{U} = \{U_i\}$, clustering $\mathcal{C}$

1: **for** ogni $U_i \in \mathcal{U}$ **do**
2: Calcola $V_i = f^{-1}(U_i) = \{x \in \mathcal{X} : f(x) \in U_i\}$
3: Applica $\mathcal{C}(V_i)$ ottenendo cluster $\{C_{i,1}, C_{i,2}, \ldots\}$
4: **end for**
5: Costruisci grafo: ogni $C_{i,j}$ è un nodo; collega $C_{i,j}$ e $C_{i',j'}$ se $C_{i,j} \cap C_{i',j'} \neq \emptyset$ **return** grafo G_{Mapper}

Garanzia topologica del Mapper

Sotto condizioni di buona copertura (overlap adeguato tra gli U_i) e clustering geometricamente corretto, il grafo Mapper approssima il *nervo* della copertura.

Teorema 5.3 (dei Nervi (Leray, 1945))**.** Sia $\mathcal{U} = \{U_i\}$ una copertura buona di uno spazio X (ogni intersezione finita di aperti di $\mathcal{U}$ è contrattile o vuota). Il nervo della copertura $N(\mathcal{U})$ ha la stessa omotopia di X.

Scelta della funzione di filtro per reti neurali

In applicazioni al deep learning la funzione di filtro f può essere:

- **Loss**: $f(\theta) = \mathcal{L}(\theta)$ — rivela la struttura dei bacini di attrazione;
- **Norma dei gradienti**: $f(\theta) = \|\nabla \mathcal{L}(\theta)\|$ — distingue zone piatte da zone ripide;
- **Proiezione PCA**: $f(\theta) = v_1^\top \theta$ — rivela la struttura lungo le direzioni principali di variazione dello spazio dei pesi.

§541 – Distanze tra Rappresentazioni Neurali tramite TDA

Definizione 5.8 (Firma Topologica di una Rappresentazione)**.** Sia $H \in \mathbb{R}^{n \times d}$ la matrice delle attivazioni di un layer per n input. La *firma topologica* è il diagramma di persistenza $\mathrm{Dgm}_1(H)$ costruito sul complesso di Vietoris-Rips delle n righe di H in $\mathbb{R}^d$.

Proposizione 5.1 (Invarianza della Firma Topologica)**.** La distanza bottleneck $d_\infty(\mathrm{Dgm}_1(H), \mathrm{Dgm}_1(H'))$ è invariante rispetto a trasformazioni ortogonali $H \mapsto HQ$ con $Q \in O(d)$, e Lipschitz-continua rispetto a perturbazioni dei pesi della rete (per il teorema di stabilità).

Applicazione: confronto di architetture

Dati due modelli $f^{(1)}_\theta$ e $f^{(2)}_\phi$ con rappresentazioni $H^{(1)}, H^{(2)}$ sullo stesso dataset, la distanza topologica è:

$$d_{\text{top}}(f^{(1)}, f^{(2)}) = d_\infty(\mathrm{Dgm}_1(H^{(1)}), \mathrm{Dgm}_1(H^{(2)})). \tag{5.13}$$

Questa metrica è invariante all'orientamento arbitrario dello spazio latente, al contrario della CKA (Centered Kernel Alignment) che richiede allineamento esplicito.

§542 – Persistent Homology per lo Studio del Grokking

Il fenomeno del *grokking* (Vol. VII, §444) — in cui la generalizzazione emerge tardivamente dopo un'apparente fase di overfitting — ha una firma topologica caratteristica.

Evoluzione topologica durante l'addestramento

Definendo la filtrazione sulla matrice dei pesi $W_l^{(t)}$ al passo t:

1. **Fase di memorizzazione** ($t < t_{\text{grok}}$): β_1 elevato, indicando struttura complessa e non regolare.
2. **Transizione di grokking** ($t \approx t_{\text{grok}}$): brusca diminuzione di β_1 — la rete "semplifica" la propria struttura interna.
3. **Generalizzazione** ($t > t_{\text{grok}}$): β_1 basso e stabile, indicando una rappresentazione topologicamente semplice.

Connessione con l'information bottleneck

La transizione topologica del grokking è analoga al principio dell'information bottleneck (Vol. IV, §213): durante la generalizzazione, la rete comprime la rappresentazione eliminando informazione superflua. Formalmente:

$$\beta_1(W_l^{(t)}) \downarrow \quad \Longleftrightarrow \quad I(W_l^{(t)}; \text{noise}) \downarrow, \tag{5.14}$$

dove $I(W; \text{noise})$ è l'informazione mutua tra i pesi e il rumore di training. La compressione topologica (riduzione di β_1) corrisponde a una riduzione dell'informazione sul rumore, ovvero a una migliore generalizzazione.

Transizioni di fase topologiche

L'evoluzione di β_1 durante l'addestramento può essere vista come una *transizione di fase topologica*: il sistema passa da una fase "disorientata" (alta β_1, analoga alla fase paramagnética nella fisica statistica) a una fase "ordinata" (bassa β_1, analoga alla fase ferromagnetica). La temperatura della transizione corrisponde all'inverso della norma dei pesi.

Capitolo 6

Approfondimenti e Connessioni Trasversali

6.1 Teoria dell'Informazione nei Sistemi Agentici

Informazione di Fisher e ottimalità dell'aggiornamento della belief

L'aggiornamento della belief state (1.1) è, dal punto di vista dell'informazione di Fisher, l'aggiornamento Bayesiano ottimale: minimizza la *perdita di informazione* nella transizione dalla prior b_t alla posterior b_{t+1}.

Proposizione 6.1 (Aggiornamento Bayesiano come contrazione della Fisher)**.** Sia $I_F(b)$ l'informazione di Fisher della belief state $b \in \Delta(S)$ rispetto allo stato vero s^*. L'aggiornamento Bayesiano $b \mapsto b_{a,o}$ soddisfa:

$$D_{\mathrm{KL}}(b_{a,o} \| b) \leq D_{\mathrm{KL}}(b \| b_{\text{uniform}}) - D_{\mathrm{KL}}(b_{a,o} \| b_{\text{uniform}}), \tag{6.1}$$

ovvero ogni aggiornamento Bayesiano *non aumenta* la distanza dall'uniforme: la belief si concentra man mano che si accumulano osservazioni.

Entropia della catena di pensiero e lunghezza ottimale

Sia $|r|$ la lunghezza (in token) del pensiero generato da un agente ReAct. Il trade-off tra qualità del pensiero e costo computazionale si formalizza come:

$$\max_{|r| \leq L} [H(a_t|c_t) - H(a_t|c_t, r)] - \lambda \cdot |r|, \tag{6.2}$$

dove λ è il costo per token. La soluzione ottimale $|r|^*$ bilancia il guadagno informativo del pensiero con il suo costo. Per pensieri che seguono una struttura autoregressiva, la riduzione dell'entropia per token aggiuntivo decresce geometricamente:

$$H(a_t|c_t, r_{1:l+1}) - H(a_t|c_t, r_{1:l}) \approx \alpha \cdot \rho^l, \quad \rho \in (0, 1), \tag{6.3}$$

il che giustifica l'esistenza di una lunghezza ottimale finita $|r|^* = \log(1/\lambda)/\log(1/\rho)$.

Regret matching e convergenza agli equilibri correlati

Algorithm 3 Regret Matching (Hart e Mas-Colell, 2000)

Require: Giocatore i, azioni A_i, orizzonte T

Inizializza $R_i^0(a) = 0$ per ogni $a \in A_i$

for $t = 1, \dots, T$ **do**

  Gioca $a_t^i \sim \sigma_t^i$, dove $\sigma_t^i(a) \propto \max(R_{t-1}^i(a), 0)$ (o uniforme se tutti ≤ 0)

  Osserva a_t^{-i} e aggiorna il regret: $R_t^i(a) = R_{t-1}^i(a) + [U_i(a, a_t^{-i}) - U_i(a_t^i, a_t^{-i})]$

end forreturn $\bar{\sigma}^i = \frac{1}{T}\sum_{t=1}^{T} \sigma_t^i$ (strategia media)

Teorema 6.1 (Convergenza del Regret Matching)**.** Con il regret matching, il regret esterno medio soddisfa:

$$\frac{1}{T}\max_{a \in A_i} \sum_{t=1}^{T} [U_i(a, a_t^{-i}) - U_i(a_t^i, a_t^{-i})] \leq \frac{\Delta_i}{\sqrt{T}}, \tag{6.4}$$

dove $\Delta_i = \max_{a,b,a^{-i}} |U_i(a, a^{-i}) - U_i(b, a^{-i})|$ è l'ampiezza delle utilità. Di conseguenza, la distribuzione empirica delle azioni congiunte converge all'insieme degli equilibri correlati in $O(1/\sqrt{T})$.

Dimostrazione. [Sketch] Il regret cumulativo di azione a è $R_T^i(a) = \sum_t [U_i(a, a_t^{-i}) - U_i(a_t^i, a_t^{-i})]$. Con regret matching, il regret normalizzato $R_T^i(a)/T \leq \Delta_i/\sqrt{T}$ per il lemma di Azuma-Hoeffding applicato a differenze martingale limitate da Δ_i. Un profilo con regret esterno ≤ 0 per ogni giocatore è, per definizione, un equilibrio correlato: quindi l'insieme dei limiti delle distribuzioni empiriche è contenuto nell'insieme degli equilibri correlati. □

6.2 Analisi Spettrale dei Modelli di Generazione Video

Spettro temporale e aliasing

Un video a frequenza di campionamento f_s (frame al secondo) può rappresentare frequenze temporali fino a $f_s/2$ (frequenza di Nyquist). I modelli di generazione video devono implicitamente imparare a rispettare questo limite: movimenti più veloci della frequenza di Nyquist producono aliasing.

Nel dominio frequenziale, un frame video $v_t \in \mathbb{R}^{H \times W \times 3}$ ha trasformata di Fourier 2D:

$$\hat{v}_t(u, v) = \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} v_t(h, w)\, e^{-2\pi i (uh/H + vw/W)}. \tag{6.5}$$

La trasformata spazio-temporale del video completo è:

$$\hat{V}(u, v, \omega) = \sum_{t=0}^{T-1} \hat{v}_t(u, v)\, e^{-2\pi i \omega t/T}, \tag{6.6}$$

dove $\omega \in \{0, \dots, T-1\}$ è la frequenza temporale. I modelli di attenzione spaziotemporale fattorizzata (3.9) corrispondono ad applicare trasformazioni separabili in u, v (spaziale) e in ω (temporale), perdendo le correlazioni spazio-temporali incrociate.

Analisi della compressione con 3D-VAE

La qualità della compressione del 3D-VAE si misura tramite il *PSNR* (Peak Signal-to-Noise Ratio) e la *distorsione di rata* (rate-distortion):

$$R(\hat{D}) = \min_{q_\phi : \mathbb{E}[\|v-\hat{v}\|_2^2] \leq \hat{D}} I(v; z), \tag{6.7}$$

dove $I(v; z)$ è l'informazione mutua tra il video v e il suo latente z. Per un VAE gaussiano con fattore di compressione $s_T s_S^2$ e dimensione del latente d_z:

$$I(v; z) \leq \frac{d_z \cdot T_z H_z W_z}{2} \log\left(1 + \frac{\text{SNR}}{s_T s_S^2}\right), \tag{6.8}$$

dove SNR è il rapporto segnale-rumore del video originale. Il trade-off qualità-compressione è quindi governato dal prodotto $d_z \cdot T_z H_z W_z$ (dimensione totale del latente).

6.3 Analisi di Complessità degli Algoritmi di Continual Learning

Tabella comparativa dei metodi

Tabella 6.1: Confronto tra i principali metodi di continual learning trattati nel Volume IX.

Metodo	**Mem.**	**Comp./step**	**Forgetting**	**Garanzia**
IRL (naive)	$O(P)$	$O(P)$	Catastrofico	Nessuna
EWC	$O(P)$	$O(P)$	Basso	MAP approx.
Online EWC	$O(P)$	$O(P)$	Basso	MAP approx.
SI	$O(P)$	$O(P)$	Basso	Euristica
PNN	$O(KP)$	$O(KP)$	Zero	Architetturale
PackNet	$O(P)$	$O(P)$	Zero	Architetturale
GEM	$O(KM)$	$O(KP)$	Zero	Per-task
A-GEM	$O(M)$	$O(P)$	Basso	In media

Legenda: P = numero di parametri, K = numero di task, M = dimensione buffer.

iCaRL: Incremental Classifier and Representation Learning

iCaRL (Rebuffi et al., 2017) è un metodo di replay che combina un classificatore *nearest-mean* (nel feature space) con un buffer di *exemplar* per task:

L'idea chiave è che gli exemplar selezionati approssimano il prototipo della classe: $\frac{1}{|P_c|}\sum_{x \in P_c} f_\theta(x) \approx \mu_c$. La classificazione al test time usa il nearest-mean: $\hat{y} = \arg\min_c \|f_\theta(x) - \mu_c\|$.

Algorithm 4 iCaRL: aggiornamento del buffer degli exemplar

Require: Encoder f_θ, nuovo task con classe c, budget m per classe

Calcola il prototipo della classe: $\mu_c = \frac{1}{|X_c|} \sum_{x \in X_c} f_\theta(x)$
Inizializza exemplar set $P_c \leftarrow \emptyset$
for $k = 1, \ldots, m$ **do**
    $x^* \leftarrow \arg\min_{x \in X_c \setminus P_c} \left\| \mu_c - \frac{1}{k} \left(f_\theta(x) + \sum_{x' \in P_c} f_\theta(x') \right) \right\|$
    $P_c \leftarrow P_c \cup \{x^*\}$
end forreturn P_c

Proposizione 6.2 (Errore del nearest-mean con exemplar)**.** Sia $\epsilon_m = \left\| \mu_c - \frac{1}{m} \sum_{x \in P_c} f_\theta(x) \right\|$ l'errore del prototipo approssimato. L'errore di classificazione del nearest-mean è:

$$P(\hat{y} \neq y) \leq P\left(\min_{c' \neq y} \|f_\theta(x) - \mu_{c'}\| \leq \|f_\theta(x) - \mu_y\| + 2\epsilon_m \right). \tag{6.9}$$

Con il greedy selection di iCaRL, $\epsilon_m \leq \|\mu_c\|_2 / m$ (bound del caso peggiore).

6.4 Applicazioni della TDA alle Architetture Transformer

Topologia degli attention head

Gli attention head in un Transformer (Vol. II, §56–60) producono distribuzioni di attenzione $A \in \mathbb{R}^{n \times n}$ (matrice $n \times n$ di pesi di attenzione per n token). La struttura topologica di A può essere analizzata tramite TDA:

Definizione 6.1 (Grafo di Attenzione)**.** Per un attention head con matrice A, il *grafo di attenzione* è il grafo pesato $G_A = (V, E, w)$ con $V = \{1, \ldots, n\}$ (i token), $E = V \times V$, e pesi $w(i, j) = A_{ij}$.

La persistent homology del grafo di attenzione (usando la filtrazione per soglia sui pesi) rivela:

- $\beta_0(\varepsilon)$: numero di componenti connesse al livello di attenzione ε — misura la *globalità* dell'attenzione.
- $\beta_1(\varepsilon)$: numero di cicli — misura la presenza di strutture ricorrenti nel pattern di attenzione.

Diagramma di persistenza delle rappresentazioni layerwise

Definendo $H_l \in \mathbb{R}^{n \times d}$ come le rappresentazioni al layer l, la *traiettoria topologica* del Transformer è la sequenza di diagrammi:

$$\mathrm{Dgm}_1(H_0), \mathrm{Dgm}_1(H_1), \ldots, \mathrm{Dgm}_1(H_L). \tag{6.10}$$

Studi empirici (Magai, 2022) mostrano che:

1. Nei primi layer, β_1 è basso: le rappresentazioni sono locali (simili agli embedding di input).
2. Nei layer intermedi, β_1 aumenta: si formano strutture globali complesse corrispondenti alla sintassi e semantica.
3. Nei layer finali, β_1 decresce: le rappresentazioni si specializzano per il task specifico.

Questa traiettoria topologica è *specifica dell'architettura*: BERT e GPT-2 mostrano profili distinti, nonostante usino la stessa architettura Transformer di base.

Omologia persistente e generalizzazione

La connessione tra topologia delle rappresentazioni e generalizzazione si formalizza tramite il seguente risultato:

Proposizione 6.3 (Complessità topologica e bound di generalizzazione)**.** Sia f_θ una rete neurale e $H \in \mathbb{R}^{n\times d}$ la matrice delle sue rappresentazioni su n training points. Il bound di generalizzazione PAC-Bayes (Vol. VI, §347) ammette la seguente variante topologica:

$$\mathcal{L}_{\text{test}}(f_\theta) \leq \mathcal{L}_{\text{train}}(f_\theta) + O\left(\sqrt{\frac{\text{pers}_1(H)\cdot \log n}{n}}\right), \tag{6.11}$$

dove $\text{pers}_1(H) = \sum_{(b,d)\in \text{Dgm}_1(H)} (d-b)^2$ è la *persistenza totale* di ordine 2.

L'interpretazione è che rappresentazioni topologicamente semplici (bassa persistenza totale) tendono a generalizzare meglio: la rete non ha "memorizzato" strutture topologiche complesse specifiche del training set.

6.5 Scaling Laws Unificate: Training-Time e Test-Time

Il modello unificato di compute

Le scaling laws di Chinchilla (Vol. VIII, §451) caratterizzano la loss di training:

$$\mathcal{L}(N, D) = A\, N^{-\alpha} + B\, D^{-\beta} + \mathcal{L}_\infty, \tag{6.12}$$

dove N è il numero di parametri e D è il numero di token di training. Il test-time compute scaling law (Sezione B, §519) caratterizza la performance in inferenza:

$$\text{Err}(C_{\text{test}}) = C_0 \cdot C_{\text{test}}^{-\alpha_{\text{test}}}. \tag{6.13}$$

Un modello unificato deve descrivere la performance come funzione del *budget totale* $C_{\text{tot}} = C_{\text{train}} + n_{\text{query}} \cdot C_{\text{test}}$:

Proposizione 6.4 (Ottimizzazione del budget totale)**.** Per un budget totale C_{tot} e n_q query, la ripartizione ottimale training/test è:

$$C^*_{\text{train}} = \frac{\alpha_{\text{train}}}{\alpha_{\text{train}} + \alpha_{\text{test}}} \cdot C_{\text{tot}}, \qquad C^*_{\text{test}} = \frac{\alpha_{\text{test}}}{\alpha_{\text{train}} + \alpha_{\text{test}}} \cdot \frac{C_{\text{tot}}}{n_q}. \tag{6.14}$$

Dimostrazione. Minimizzare $\mathcal{L}(C_{\text{train}}) + \text{Err}(C_{\text{test}})$ soggetto a $C_{\text{train}} + n_q C_{\text{test}} = C_{\text{tot}}$. Le condizioni del primo ordine danno: $\alpha_{\text{train}} C'_0 C_{\text{train}}^{-\alpha_{\text{train}}-1} = n_q \alpha_{\text{test}} C_1 C_{\text{test}}^{-\alpha_{\text{test}}-1}$, da cui si ricava il rapporto $C_{\text{train}}/C_{\text{test}}$ e si sostituisce nel vincolo. □

Implicazioni pratiche

Per $\alpha_{\text{train}} \approx \alpha_{\text{test}}$ (scaling comparabile), la ripartizione ottimale è 50/50. Per $\alpha_{\text{test}} > \alpha_{\text{train}}$ (il test-time compute scala meglio del training), conviene investire più nel test-time compute: questo è il regime dei modelli "thinking" su problemi difficili.

Il fattore n_q è critico: per un modello interrogato molte volte ($n_q \gg 1$), quasi tutto il budget dovrebbe andare al training. Per un modello usato poche volte su problemi molto difficili (n_q piccolo), il test-time compute è più efficiente.

6.6 Prove di Convergenza Estese

Convergenza di MAPPO in ambienti cooperativi

Teorema 6.2 (Convergenza di MAPPO, caso cooperativo)**.** In un gioco cooperativo con $R_i = R$ per ogni i, MAPPO con critici centralizzati converge a un punto stazionario della funzione obiettivo globale:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\pi_\theta}}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \mathbf{a}_t)\right] \tag{6.15}$$

sotto le ipotesi standard di PPO (gradiente Lipschitz, step-size decrescente, buffer sufficientemente grande).

Dimostrazione. [Sketch] Nel caso cooperativo, $J(\boldsymbol{\theta}) = \sum_i J_i(\boldsymbol{\theta})$ (per linearità dell'aspettativa, essendo $R_i = R$). La loss MAPPO (1.18) è il surrogate lower bound di PPO applicato congiuntamente a tutti gli agenti. Poiché l'obiettivo congiunto è separabile nei parametri θ_i (ogni agente ottimizza la propria politica con il critico centralizzato come input), il teorema di convergenza di PPO (Schulman et al., 2017, Theorem 1) si applica separatamente a ciascun agente, con la differenza che il vantaggio $\hat{A}^i_t$ usa il ritorno globale R invece del ritorno individuale. La convergenza al punto stazionario di J segue dall'argomento del monotone improvement di PPO. □

EWC multi-task: bound di forgetting

Proposizione 6.5 (Bound di forgetting con EWC)**.** Sia $\Delta\theta = \theta - \theta_1^*$ la perturbazione dei parametri dopo l'addestramento su task 2 con EWC. Il forgetting è bounded:

$$\mathcal{L}_1(\theta) - \mathcal{L}_1(\theta_1^*) \leq \frac{1}{2}\Delta\theta^\top F_1 \Delta\theta \leq \frac{\mathcal{L}_2(\theta_1^*) - \mathcal{L}_2(\theta_2^*)}{\lambda}, \tag{6.16}$$

dove la seconda disuguaglianza usa il fatto che EWC minimizza $\mathcal{L}_2(\theta) + \frac{\lambda}{2}\Delta\theta^\top \mathrm{diag}(F_1)\Delta\theta$.

Dimostrazione. La prima disuguaglianza è l'approssimazione di Taylor (4.3). Per la seconda: all'ottimo di EWC, $\mathcal{L}_2(\theta) + \frac{\lambda}{2}\Delta\theta^\top F_1 \Delta\theta \leq \mathcal{L}_2(\theta_1^*)$ (il valore al punto di partenza), da cui $\frac{\lambda}{2}\Delta\theta^\top F_1 \Delta\theta \leq \mathcal{L}_2(\theta_1^*) - \mathcal{L}_2(\theta) \leq \mathcal{L}_2(\theta_1^*) - \mathcal{L}_2(\theta_2^*)$. Quindi $\Delta\theta^\top F_1 \Delta\theta \leq 2[\mathcal{L}_2(\theta_1^*) - \mathcal{L}_2(\theta_2^*)]/\lambda$. □

Il bound mostra che il forgetting è controllato da: (i) la forza della regolarizzazione λ; (ii) il guadagno $\mathcal{L}_2(\theta_1^*) - \mathcal{L}_2(\theta_2^*)$ (quanto è difficile il task 2 se si parte da θ_1^*). Se i task sono simili (piccolo guadagno), il forgetting è automaticamente limitato.

6.7 Esempi Computazionali Espliciti

Omologia persistente: un esempio numerico

Consideriamo 4 punti nel piano: $x_1 = (0,0)$, $x_2 = (1,0)$, $x_3 = (1,1)$, $x_4 = (0,1)$ (i vertici di un quadrato).

Filtrazione di Vietoris-Rips:

- $\varepsilon = 0$: solo 4 vertici (0-simplessi). $\beta_0 = 4$, $\beta_1 = 0$.
- $\varepsilon = 1$: si aggiungono i 4 lati del quadrato (distanza 1). $\beta_0 = 1$, $\beta_1 = 1$ (il quadrato è un ciclo).
- $\varepsilon = \sqrt{2}$: si aggiungono le due diagonali (distanza $\sqrt{2}$). I due triangoli $[x_1, x_2, x_3]$ e $[x_1, x_3, x_4]$ riempiono il ciclo. $\beta_0 = 1$, $\beta_1 = 0$.

Il diagramma di persistenza ha un solo punto: $(b, d) = (1, \sqrt{2})$, con persistenza $\sqrt{2} - 1 \approx 0.41$. Questo punto rappresenta il "buco" del quadrato che nasce a $\varepsilon = 1$ e muore a $\varepsilon = \sqrt{2}$.

QMIX: un esempio con due agenti

Consideriamo un gioco cooperativo con 2 agenti, 2 azioni ciascuno, e la seguente matrice di payoff:

$$R = \begin{pmatrix} 8 & -12 \\ -12 & 6 \end{pmatrix}, \tag{6.17}$$

dove R_{ij} è la ricompensa congiunta quando l'agente 1 gioca azione i e l'agente 2 gioca azione j. L'azione ottimale congiunta è $(1,1)$ (payoff 8).

Supponiamo che i Q-value individuali siano $Q_1(1) = 5, Q_1(2) = 3$ e $Q_2(1) = 4, Q_2(2) = 2$. Con QMIX, $Q_{\text{tot}} = f_s(Q_1, Q_2)$ con pesi positivi $w_1 = w_2 = 1$:

$$\begin{aligned} Q_{\text{tot}}(1,1) &= f_s(5,4) = 9, \\ Q_{\text{tot}}(1,2) &= f_s(5,2) = 7, \\ Q_{\text{tot}}(2,1) &= f_s(3,4) = 7, \\ Q_{\text{tot}}(2,2) &= f_s(3,2) = 5. \end{aligned}$$

Il principio IGM predice $\arg\max_{\mathbf{a}} Q_{\text{tot}} = (1,1)$, che coincide con l'azione ottimale congiunta. In questo caso la fattorizzazione QMIX è sufficiente.

Nota. QMIX non può rappresentare tutti i giochi cooperativi. Se la matrice ottimale richiedesse $Q_{\text{tot}}(1,2) > Q_{\text{tot}}(1,1)$ pur con $Q_1(1) > Q_1(2)$, la monotonicità verrebbe violata. QMIX fallisce su questi giochi: è necessario usare QPLEX (Wang et al., 2020) che ammette fattorizzazioni più espressive.

EWC: calcolo della Fisher diagonale

Consideriamo una rete lineare $f_\theta(x) = \theta^\top x$ con $x \in \mathbb{R}^d$ e loss quadratica $\mathcal{L}_1(\theta) = \mathbb{E}[(y - \theta^\top x)^2]/2$. Il log-likelihood è $\log P(y|x,\theta) = -\frac{1}{2\sigma^2}(y - \theta^\top x)^2 + \text{const}$.

La Fisher diagonale è:

$$F_{1,jj} = \mathbb{E}_x\left[\left(\frac{\partial \log P}{\partial \theta_j}\right)^2\right] = \mathbb{E}_x\left[\left(\frac{y - \theta^\top x}{\sigma^2} x_j\right)^2\right] = \frac{1}{\sigma^4}\,\text{Var}(y - \theta^\top x)\,\mathbb{E}[x_j^2]. \tag{6.18}$$

Se $y - \theta^\top x \sim \mathcal{N}(0, \sigma^2)$ (residui gaussiani), allora $F_{1,jj} = \mathbb{E}[x_j^2]/\sigma^2$. La regolarizzazione EWC penalizza i parametri θ_j che corrispondono a feature x_j con alta varianza: questi sono i parametri "importanti" per il task 1.

6.8 Connessioni tra le Sezioni del Volume

Il PRM come funzione valore del POMDP agentivo

La connessione più profonda del Volume IX è tra i Process Reward Models (Sezione B, §521) e il framework POMDP degli agenti (Sezione A, §511). Formalmente:

Proposizione 6.6 (Equivalenza PRM-POMDP)**.** In un agente che risolve un problema di ragionamento x, la traiettoria di pensiero $r_1, r_2, \ldots, r_t$ è esattamente una sequenza di osservazioni nel POMDP del problema. Il PRM $V_\phi^{\text{PRM}}(x, r_{\le t})$ è la funzione valore del POMDP condizionata sulla belief state b_t indotta da $r_{\le t}$:

$$V_\phi^{\text{PRM}}(x, r_{\le t}) = V^*(b_t) = P(R = 1 \mid x, r_{\le t}). \tag{6.19}$$

MCTS con PRM è quindi la pianificazione nel POMDP del ragionamento: la selezione UCB1 bilancia esplorazione (provare nuove catene di pensiero) e sfruttamento

(continuare le catene più promettenti).

EWC come approssimazione dell'omologia persistente

Un'altra connessione inaspettata collega EWC (Sezione D) alla topologia delle loss (Sezione E). La matrice di Fisher diagonale $F_{1,jj}$ misura la curvatura della loss di task 1 lungo l'asse j: parametri con alta Fisher corrispondono a direzioni di alta curvatura, che nella filtrazione di livello della loss (4.3) sono le direzioni che "alzano" la loss più rapidamente.

Formalmente, i parametri con $F_{1,jj} > c$ (per una soglia c) corrispondono alle direzioni in cui la loss di task 1 è c-sensibile: restringere il movimento in queste direzioni equivale a rimanere nell'intorno del minimo di task 1 dove $\mathcal{L}_1(\theta) \leq c$. La geometria topologica di questo intorno è precisamente catturata dalla filtrazione di livello $\mathcal{L}_1^{-1}(-\infty, c]$ della Sezione E.

Grokking come transizione di fase nel continual learning

Il grokking (Sezione E, §542) può essere reinterpretato come un caso speciale di continual learning in cui il modello deve "apprendere" due task sequenziali:

1. **Task 1**: memorizzazione del training set (overfitting).
2. **Task 2**: apprendimento della struttura algoritmica generalizzante.

La transizione topologica a $t = t_{\text{grok}}$ corrisponde a un "forgetting" positivo del task 1 (memorizzazione) a favore del task 2 (generalizzazione). Questo è un caso in cui il forgetting è *desiderabile*: la rete dimentica la soluzione memorizzata e converge alla soluzione algoritmica.

Il framework EWC applicato al grokking suggerisce che la transizione è possibile solo quando la Fisher del task 1 (memorizzazione) è *bassa* nelle direzioni del task 2 (algoritmo): ovvero quando la soluzione algoritmica non interferisce con la memorizzazione precedente. Questa è esattamente la condizione geometrica che rende il grokking possibile.

Conclusioni e Prospettive

Il Volume IX ha completato il ciclo con cinque aree fondamentali rimaste scoperte nei volumi precedenti. Ripercorriamo i contributi principali e le connessioni trasversali emerse.

AI Agentiva (Sezione A). La formalizzazione degli agenti LLM come POMDP ha rivelato che il pensiero esplicito è matematicamente equivalente a ridurre l'entropia dell'azione condizionata all'osservazione. La struttura lineare a tratti di V^* chiarisce perché gli LLM con contesto finito possono approssimare la funzione valore ottimale. Il MARL ha mostrato che la teoria dei giochi — Nash, Shapley, giochi a potenziale — fornisce il linguaggio corretto per sistemi multi-agente. Il collegamento con il RL del Volume III è diretto: MAPPO è PPO con critico centralizzato. La novità rispetto alla versione breve è il trattamento degli *equilibri correlati* e dei *mean-field games*, che colmano il gap tra la teoria dei giochi a due agenti e i sistemi a molti agenti reali.

Inference-Time Compute (Sezione B). MCTS con PRM e best-of-N rappresentano un secondo asse di scaling, ortogonale alla dimensione del modello. Il collegamento matematico più profondo è con il POMDP della Sezione A: il PRM è la funzione valore della belief state, e MCTS è la pianificazione nel POMDP del ragionamento. La derivazione formale di GRPO dal principio di massimizzazione vincolata chiarisce il ruolo della normalizzazione del vantaggio come stima di baseline.

Generazione Video (Sezione C). Il Video Diffusion Model è DDPM (Vol. III) con attenzione spaziotemporale fattorizzata. L'architettura Sora estende il DiT (Vol. VII) a patch 3D con RoPE tridimensionale. La derivazione del *flow matching* per video e l'analisi comparativa delle architetture (VDM, Sora, VideoGPT, CM) forniscono un quadro unificato della generazione video moderna.

Continual Learning (Sezione D). EWC è un'approssimazione al posteriore MAP sequenziale con Fisher diagonale. GEM garantisce matematicamente il non-peggioramento tramite proiezione sul poliedro dei gradienti compatibili. Le nuove aggiunte — Online EWC, Synaptic Intelligence e A-GEM — completano il panorama con metodi che bilanciano efficacia computazionale e garanzie teoriche.

Topologia Applicata (Sezione E). L'omologia persistente fornisce invarianti topologici dei loss landscape e delle rappresentazioni neurali stabili e calcolabili. Le nuove sezioni sulla persistenza vettoriale (persistence images e landscapes), sulla distanza di Wasserstein tra diagrammi, sulle fasi topologiche dell'addestramento e sulla connessione grokking-information bottleneck ampliano significativamente il framework e le sue applicazioni pratiche.

Prospettive per un Volume X. Restano alcune aree con fondamenti matematici rigorosi che meriterebbero trattamento sistematico:

- *Fisica statistica e spin glass theory*: il metodo delle repliche per calcolare la loss di training di equilibrio nei trasformatori;

- *Rappresentazioni 3D avanzate*: 3D Gaussian Splatting (3DGS) con differenziabilità della rasterizzazione gaussiana;
- *Active Learning e Bayesian Experimental Design*: uncertainty sampling, query-by-committee, expected information gain;
- *AI per la matematica formale*: integrazione LLM+Lean/Isabelle e program synthesis differenziabile.

Capitolo 7

Derivazioni Avanzate e Metodi di Secondo Ordine

7.1 Ottimizzazione Naturale del Gradiente nel MARL

Geometria di Fisher-Rao nello spazio delle politiche

Nel MARL, lo spazio delle politiche $\Delta(A_i)$ per ciascun agente è un simplesso, dotato naturalmente della metrica di Fisher-Rao (Vol. IV, §201–206). Il gradiente naturale per l'agente i è:

$$\tilde{\nabla}_{\theta_i} J_i = F_{\theta_i}^{-1} \nabla_{\theta_i} J_i, \tag{7.1}$$

dove $F_{\theta_i} \in \mathbb{R}^{d_i \times d_i}$ è la matrice di Fisher della politica π_{θ_i}:

$$F_{\theta_i} = \mathbb{E}_{s,a_i \sim \pi_{\theta_i}} \left[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i|s) \, \nabla_{\theta_i} \log \pi_{\theta_i}(a_i|s)^\top \right]. \tag{7.2}$$

Il gradiente naturale è invariante rispetto a reparametrizzazioni di π_{θ_i}: se $\theta_i' = g(\theta_i)$ è un cambio di parametrizzazione, il gradiente naturale nella nuova coordinata è identico a quello nella vecchia coordinata, trasformato in modo covariante. Questa proprietà è cruciale per la convergenza uniforme nel MARL: ogni agente ottimizza nella stessa metrica indipendentemente dalla sua parametrizzazione.

Teorema 7.1 (Natural Policy Gradient nel MARL)**.** In un gioco cooperativo con politiche softmax $\pi_{\theta_i}(a_i|s) \propto e^{\theta_i^\top \phi(s,a_i)}$, il gradiente naturale per ciascun agente i è:

$$\tilde{\nabla}_{\theta_i} J_i = \mathbb{E}_\pi[Q^{\pi_i}(s,a_i)\, \phi(s,a_i)], \tag{7.3}$$

dove $Q^{\pi_i}(s,a_i) = \mathbb{E}[\sum_t \gamma^t R_i | s_0 = s, a_0^i = a_i, \pi]$ è la funzione Q dell'agente i, e $\phi(s,a_i)$ sono le feature della politica. Questo risultato segue dal teorema del gradiente naturale per politiche tabellari (Kakade, 2001).

Fisher Information Matrix per le politiche Transformer

Per politiche parametrizzate da un Transformer con milioni di parametri, la Fisher completa è computazionalmente inaccessibile. Si usano le seguenti approssimazioni:

K-FAC per politiche (Kronecker-Factored Approximate Curvature): per un layer lineare $W \in \mathbb{R}^{m \times n}$ della politica, si fattorizza la Fisher come:

$$F_W \approx A \otimes G, \qquad A = \mathbb{E}[a\, a^\top] \in \mathbb{R}^{n \times n}, \quad G = \mathbb{E}[g\, g^\top] \in \mathbb{R}^{m \times m}, \tag{7.4}$$

dove $a \in \mathbb{R}^n$ è l'attivazione in input al layer e $g \in \mathbb{R}^m$ è il gradiente dell'output. L'inversione è: $F_W^{-1} \approx A^{-1} \otimes G^{-1}$, riducendo il costo da $O((mn)^3)$ a $O(n^3 + m^3)$.

Analisi della convergenza di QMIX con gradiente naturale

Proposizione 7.1 (QMIX con Natural Gradient converge più velocemente)**.** Sia $\epsilon_t = \mathcal{L}_{\text{QMIX}}(\theta_t) - \mathcal{L}_{\text{QMIX}}(\theta^*)$ l'errore di training al passo t. Con learning rate η e gradiente naturale:

$$\epsilon_{t+1} \leq (1 - \eta\, \lambda_{\min}(F_{\theta_t}))\, \epsilon_t, \tag{7.5}$$

contro il tasso di convergenza del gradiente euclideo:

$$\epsilon_{t+1} \leq \left(1 - \eta\, \frac{\lambda_{\min}(F_{\theta_t})}{\lambda_{\max}(F_{\theta_t})}\right) \epsilon_t. \tag{7.6}$$

Il rapporto $\kappa = \lambda_{\max}/\lambda_{\min}$ (condition number della Fisher) può essere 10^3–10^6 per reti neurali profonde, giustificando il vantaggio del gradiente naturale.

7.2 Teoria delle Approssimazioni Funzionali per il POMDP

Point-Based Value Iteration (PBVI)

La soluzione esatta del POMDP (calcolare la funzione V^* lineare a tratti) richiede un numero esponenziale di piani α nell'orizzonte. PBVI (Pineau et al., 2003) approssima V^* mantenendo un insieme $\mathcal{B}$ di belief state rappresentative:

Algorithm 5 Point-Based Value Iteration

Require: insieme di belief $\mathcal{B} = \{b_1, \ldots, b_m\}$, tolleranza ε

1: Inizializza $\Gamma^0 = \{\alpha_{\text{pessimistic}}\}$ (piani α pessimistici)
2: **repeat**
3: **for** ogni $b \in \mathcal{B}$ **do**
4: Per ogni azione a e osservazione o, calcola: $\alpha_a(s) = R(s,a) + \gamma \sum_{s'} T(s'|s,a) \max_{\alpha \in \Gamma} \sum_{s''} Z(o|s',a)\alpha(s'')$
5: Seleziona il piano $\alpha^* = \arg\max_{\alpha_a} b^\top \alpha_a$
6: Aggiungi α^* a Γ^{t+1}
7: **end for**
8: **until** $\max_b |V^{t+1}(b) - V^t(b)| \leq \varepsilon$

Teorema 7.2 (Complessità di PBVI)**.** PBVI con m belief state e K passi ha complessità:

$$O(K \cdot m \cdot |A| \cdot |O| \cdot |\Gamma| \cdot |S|^2), \tag{7.7}$$

contro $O(|S|^{|O|\cdot|A|})$ dell'algoritmo esatto (esponenziale in $|O| \cdot |A|$). La qualità dell'approssimazione dipende dalla copertura di $\mathcal{B}$: se $\mathcal{B}$ è δ-densa nello spazio delle belief, allora $\left\|V^* - V^{\text{PBVI}}\right\|_\infty \leq \delta L_V$, dove L_V è la costante di Lipschitz di V^*.

Approssimazione del POMDP con reti ricorrenti

Per POMDP con spazi di stato grandi (come gli agenti LLM), la belief state è approssimata con una rete ricorrente (RNN o Transformer):

$$h_t = f_\phi(h_{t-1}, o_t, a_{t-1}), \qquad \hat{b}_t \approx g_\phi(h_t), \tag{7.8}$$

dove $h_t \in \mathbb{R}^d$ è lo stato nascosto e g_ϕ mappa lo stato nascosto alla belief approssimata. La loss di addestramento combina:

$$\mathcal{L}_{\text{POMDP-RNN}} = \underbrace{\mathbb{E}\left[\left\|\hat{b}_t - b_t\right\|_2^2\right]}_{\text{belief accuracy}} + \underbrace{\mathbb{E}\left[(V(h_t) - V^*(b_t))^2\right]}_{\text{valore accuracy}}. \tag{7.9}$$

Nel contesto degli LLM, h_t corrisponde al KV-cache (Key-Value cache): la sequenza di token nel contesto funge da memoria esplicita della belief state, con la funzione di transizione f_ϕ implementata dall'attenzione causale.

7.3 Modelli di Diffusione: Connessioni Profonde

Score matching e DDPM: derivazione unificata

Il Video Diffusion Model (Sezione C, §526) si inquadra nella teoria generale del *score matching* (Hyvärinen, 2005). Il *score* di una distribuzione $p(v)$ è $\nabla_v \log p(v)$. Il denoiser ottimale $\epsilon^*_\theta(v_k, k)$ è collegato allo score tramite:

$$\epsilon^*_\theta(v_k, k) = -\sqrt{1 - \bar{\alpha}_k}\, \nabla_{v_k} \log p_k(v_k), \tag{7.10}$$

dove $p_k(v_k) = \int q(v_k|v_0) p_{\text{data}}(v_0)\, dv_0$ è la distribuzione marginale al passo k. Questa relazione mostra che la loss (3.9) è equivalente alla *denoising score matching loss*:

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{k, v_0, v_k}\left[\|\nabla_{v_k} \log q(v_k|v_0) - s_\theta(v_k, k)\|_2^2\right], \tag{7.11}$$

dove $s_\theta(v_k, k) \approx \nabla_{v_k} \log p_k(v_k)$ è il modello di score.

Stochastic Differential Equations per la diffusione

La formulazione più generale della diffusione usa le SDE (Song et al., 2021):

$$dv = f(v, t)\, dt + g(t)\, dW_t, \qquad t \in [0, T], \tag{7.12}$$

dove W_t è il moto Browniano standard in $\mathbb{R}^{THW}$, f è il drift e g è la diffusività. Per DDPM con schedula lineare di rumore $\beta(t)$:

$$f(v, t) = -\frac{1}{2}\beta(t)\, v, \qquad g(t) = \sqrt{\beta(t)}. \tag{7.13}$$

La SDE inversa (che campiona dalla distribuzione dei dati) è:

$$dv = \left[f(v, t) - g(t)^2\, \nabla_v \log p_t(v)\right] dt + g(t)\, d\bar{W}_t, \tag{7.14}$$

dove $\bar{W}_t$ è il moto Browniano in senso inverso. Sostituendo il modello di score s_θ si ottiene la SDE di campionamento approssimato.

Proposizione 7.2 (ODE deterministica di campionamento)**.** La SDE inversa ha un'ODE deterministica corrispondente ("probability flow ODE"):

$$\frac{dv}{dt} = f(v,t) - \frac{1}{2} g(t)^2 \, \nabla_v \log p_t(v), \tag{7.15}$$

che genera le stesse distribuzioni marginali della SDE inversa. L'ODE (7.15) può essere integrata con solver numerici (Runge-Kutta, Adams) invece dei passi DDPM, riducendo il numero di valutazioni del denoiser da 1000 a 20–50.

Conditional Flow Matching per video

Il flow matching (Lipman et al., 2022; Liu et al., 2022) costruisce un campo di velocità $u_t(v)$ che trasporta $p_0 = \mathcal{N}(0, I)$ in $p_1 = p_{\text{data}}$ lungo traiettorie rette condizionate su v_0:

$$v_t = (1-t)\,\epsilon + t\, v_0, \qquad \epsilon \sim \mathcal{N}(0, I),\ v_0 \sim p_{\text{data}}. \tag{7.16}$$

Il campo di velocità condizionale è:

$$u_t(v | v_0) = v_0 - \epsilon = \frac{v_0 - v_t}{1-t}, \tag{7.17}$$

e il modello impara il campo marginale $u_t(v) = \mathbb{E}[u_t(v|v_0)|v_t = v]$. La loss di flow matching è:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t, v_0, \epsilon}\left[\| u_\theta(v_t, t) - (v_0 - \epsilon) \|_2^2 \right]. \tag{7.18}$$

Proposizione 7.3 (Equivalenza flow matching e score matching)**.** Per la scelta $v_t = \sqrt{\bar\alpha_t}\, v_0 + \sqrt{1-\bar\alpha_t}\, \epsilon$ (interpolazione DDPM), la loss di flow matching è equivalente alla loss DDPM (3.9) a meno di una reparametrizzazione:

$$u_\theta(v_t, t) = \frac{\sqrt{\bar\alpha_t}\, v_0 - v_t}{\sqrt{1-\bar\alpha_t}} = -\frac{\sqrt{1-\bar\alpha_t}}{\sqrt{\bar\alpha_t}}\, \epsilon_\theta(v_t, t). \tag{7.19}$$

7.4 Metodi Kernel per la Persistent Homology

Kernel per diagrammi di persistenza

Per usare i diagrammi di persistenza in algoritmi kernel (SVM, Kernel PCA), si definisce un kernel positivo semi-definito su $\mathcal{D} \times \mathcal{D}$, dove $\mathcal{D}$ è lo spazio dei diagrammi. Il kernel di persistenza più usato è il *persistence weighted Gaussian kernel* (Kusano et al., 2016):

$$k_\sigma(A, B) = \sum_{p \in A} \sum_{q \in B} w(p)\, w(q)\, e^{-\|p-q\|_2^2 / (2\sigma^2)}, \tag{7.20}$$

dove $w(b, d) = \arctan(c\,(d-b)^p)$ è un peso che valorizza le feature persistenti e $\sigma > 0$ è il bandwidth del kernel.

Teorema 7.3 (Positività del persistence kernel)**.** Il persistence weighted Gaussian kernel k_σ è positivo semi-definito su $\mathcal{D}$ se $w \geq 0$ puntualmente.

Dimostrazione. Si verifica che $k_\sigma(A, B) = \langle \Phi(A), \Phi(B) \rangle_{\mathcal{H}}$ dove $\Phi(A) = \sum_{p \in A} w(p)\, \phi_\sigma(p)$ e $\phi_\sigma(p) = \mathcal{N}(\cdot; p, \sigma^2 I)$ è l'embedding nel reproducing kernel Hilbert space del kernel gaussiano. Poiché il kernel gaussiano è positivo semi-definito e la mappa Φ è lineare (con coefficienti non negativi $w(p) \geq 0$), k_σ è positivo semi-definito. □

Sliced Wasserstein per diagrammi

La distanza di Wasserstein tra diagrammi è computazionalmente costosa ($O(n^3 \log n)$ per l'algoritmo di Hungarian con n punti). La *sliced Wasserstein distance* approssima questa con:

$$\mathrm{SW}(A, B) = \int_0^\pi W_1(\pi_\theta(A), \pi_\theta(B))\, d\theta, \tag{7.21}$$

dove $\pi_\theta(A)$ è la proiezione di A sull'asse di direzione θ e W_1 è la distanza di Wasserstein 1D (calcolabile in $O(n \log n)$). L'integrale si approssima con M direzioni casuali:

$$\widehat{\mathrm{SW}}(A, B) \approx \frac{1}{M} \sum_{m=1}^{M} W_1(\pi_{\theta_m}(A), \pi_{\theta_m}(B)), \tag{7.22}$$

con convergenza $O(1/\sqrt{M})$ per la legge dei grandi numeri.

7.5 Teoria della Generalizzazione per Modelli Diffusivi

Memorizzazione vs generalizzazione nella diffusione

Un problema fondamentale per i modelli generativi è la distinzione tra *memorizzazione* (riproduzione di training samples) e *generalizzazione* (generazione di nuovi dati). Per un Video Diffusion Model addestrato su n video, il rischio di memorizzazione si misura con:

$$\mathrm{Mem}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\left[\min_{t \in [0,T]} \left\| v_\theta(t) - v_i^{(0)} \right\|_2 < \varepsilon \right], \tag{7.23}$$

dove $v_\theta(t)$ è una traiettoria di generazione e $v_i^{(0)}$ è il i-esimo training video. Empiricamente, $\mathrm{Mem}(\theta)$ cresce con la dimensione del modello e decresce con n (più dati $\Rightarrow$ meno memorizzazione).

Proposizione 7.4 (Bound di memorizzazione per DDPM)**.** Per un DDPM con N parametri addestrato su n sample con loss $\mathcal{L}_{\mathrm{VDM}} \leq \varepsilon_{\mathrm{train}}$, la probabilità di memorizzazione di un singolo campione è:

$$P(\mathrm{Mem} \geq \delta) \leq \exp\left(-\frac{n\delta}{2} + \frac{N \log(1/\varepsilon_{\mathrm{train}})}{2} \right). \tag{7.24}$$

Il modello non memorizza se $n \geq N \log(1/\varepsilon_{\text{train}})/\delta$: il numero di sample deve scalare linearmente con il numero di parametri.

Bound di FID per modelli video

Il *Fréchet Inception Distance* (FID) misura la qualità della generazione come distanza di Wasserstein-2 tra distribuzioni di feature estratte da una rete Inception:

$$\text{FID} = \|\mu_r - \mu_g\|_2^2 + \text{tr}\big(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\big)\,, \tag{7.25}$$

dove (μ_r, Σ_r) sono media e covarianza delle feature dei video reali e (μ_g, Σ_g) dei video generati. Per il FVD (Fréchet Video Distance), si usa una rete I3D invece di Inception per estrarre feature spaziotemporali.

Proposizione 7.5 (Decomposizione del FVD)**.** Il FVD si decompone in due contributi:

$$\text{FVD} = \underbrace{\|\mu_r - \mu_g\|_2^2}_{\text{bias}} + \underbrace{\text{tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})}_{\text{varianza}}\,. \tag{7.26}$$

Un modello che produce sempre lo stesso video (mode collapse) ha bias nullo ma varianza alta. Un modello che produce video casuali ha bias alto ma varianza bassa. Il FVD ottimale bilancia i due contributi.

7.6 Continual Learning con Ottimizzazione Distribuita

Federated Continual Learning

Nel *federated learning* (FL), K client addestrano un modello condiviso senza condividere i dati grezzi. Il *federated continual learning* combina FL con il continual learning: ogni client k affronta una sequenza di task locali, e deve evitare il forgetting sia locale (tra i propri task) sia globale (tra i client).

La loss federated EWC è:

$$\mathcal{L}_{\text{Fed-EWC}}(\theta) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_k(\theta) + \frac{\lambda}{2K}\sum_{k=1}^{K}\sum_{j} F_{k,jj}^{\text{local}}(\theta_j - \theta_{k,j}^*)^2, \tag{7.27}$$

dove $F_{k,jj}^{\text{local}}$ è la Fisher diagonale calcolata sui dati locali del client k e $\theta_{k,j}^*$ è il suo ottimo locale sui task precedenti.

Proposizione 7.6 (Federated EWC: bound di forgetting globale)**.** Sia $\bar{\theta}^*$ l'ottimo

globale del task precedente (aggregato con FedAvg). Il forgetting globale con Federated EWC è:

$$\frac{1}{K}\sum_k \mathcal{L}_k(\theta^{(t)}) - \frac{1}{K}\sum_k \mathcal{L}_k(\bar{\theta}^*) \leq \frac{1}{\lambda}\left[\frac{1}{K}\sum_k \mathcal{L}_k^{\text{new}}(\bar{\theta}^*) - \min_\theta \frac{1}{K}\sum_k \mathcal{L}_k^{\text{new}}(\theta)\right]. \tag{7.28}$$

Comunicazione e privacy nel Federated Continual Learning

La trasmissione della matrice di Fisher $F_k \in \mathbb{R}^P$ (anche solo la diagonale) richiede $O(P)$ comunicazione per client per round. Per $P = 10^9$ parametri e $K = 100$ client, questo è computazionalmente proibitivo. Le soluzioni includono:

1. **Sparse Fisher**: inviare solo i top-s elementi diagonali più grandi. Costo $O(s)$ per client con $s \ll P$.
2. **Random projection**: inviare $\tilde{F}_k = P^\top F_k P$ dove $P \in \mathbb{R}^{P\times r}$ è una matrice di proiezione casuale. Costo $O(r)$ con $r \ll P$.
3. **Differential privacy**: aggiungere rumore di Laplace $\text{Lap}(0, \Delta/\epsilon_{\text{priv}})$ alla Fisher per garantire ϵ_{priv}-differential privacy.

Capitolo 8

Fondamenti Probabilistici e Teoria della Misura

Questo capitolo raccoglie i fondamenti probabilistici e di teoria della misura che sottendono i risultati dei capitoli precedenti. L'obiettivo è fornire un riferimento autocontenuto per i lettori che desiderano approfondire le basi matematiche.

8.1 Processi Stocastici e Martingale

Definizioni fondamentali

Definizione 8.1 (Processo stocastico)**.** Un *processo stocastico* su uno spazio di probabilità $(\Omega, \mathcal{F}, P)$ con parametro temporale T è una famiglia di variabili aleatorie $\{X_t : t \in T\}$ a valori in uno spazio misurabile $(S, \mathcal{S})$.

Definizione 8.2 (Filtrazione e adattamento)**.** Una *filtrazione* è una famiglia crescente $(\mathcal{F}_t)_{t \geq 0}$ di σ-algebre con $\mathcal{F}_s \subseteq \mathcal{F}_t$ per $s \leq t$. Un processo (X_t) è *adattato* a $(\mathcal{F}_t)$ se X_t è $\mathcal{F}_t$-misurabile per ogni t.

Definizione 8.3 (Martingala)**.** Un processo adattato (M_t) è una *martingala* rispetto a $(\mathcal{F}_t)$ se $\mathbb{E}[|M_t|] < \infty$ e $\mathbb{E}[M_t | \mathcal{F}_s] = M_s$ per ogni $s \leq t$.

Teorema di Stopping Opzionale

Teorema 8.1 (Doob, 1953)**.** Sia $(M_t)_{t \geq 0}$ una martingala uniformemente integrabile e τ un tempo di arresto. Allora:

$$\mathbb{E}[M_\tau] = \mathbb{E}[M_0]. \tag{8.1}$$

Applicazione al MARL: il processo dei Q-value $Q_t(s, a) = \mathbb{E}[G_t | s_t = s, a_t = a]$ è una martingala rispetto alla filtrazione generata dalla traiettoria. Il teorema di stopping opzionale garantisce che il Q-value all'ottimo sia la ricompensa attesa a partire da qualunque tempo di arresto.

Disuguaglianza di Azuma-Hoeffding

Teorema 8.2 (Azuma-Hoeffding)**.** Sia $(M_k)_{k=0}^n$ una martingala con differenze $d_k = M_k - M_{k-1}$ tali che $|d_k| \le c_k$ quasi certamente. Allora per ogni $t > 0$:

$$P(M_n - M_0 \ge t) \le \exp\left(-\frac{t^2}{2\sum_{k=1}^n c_k^2}\right). \tag{8.2}$$

Questa disuguaglianza è fondamentale per l'analisi del regret di UCB1 (Lemma **??**): le differenze del processo di regret sono bounded da Δ_i (il gap tra bracci), e la disuguaglianza di Azuma-Hoeffding fornisce il bound di concentrazione necessario.

8.2 Trasporto Ottimale e Distanze tra Distribuzioni

Il problema di Kantorovich

Il trasporto ottimale formalizza la distanza tra distribuzioni in modo geometricamente naturale. Dato uno spazio metrico (X, d) e due distribuzioni $\mu, \nu \in \mathcal{P}(X)$:

Definizione 8.4 (Piano di trasporto)**.** Un *piano di trasporto* tra μ e ν è una distribuzione $\gamma \in \mathcal{P}(X \times X)$ con marginali μ e ν: $\gamma(A \times X) = \mu(A)$ e $\gamma(X \times B) = \nu(B)$. L'insieme dei piani di trasporto è $\Pi(\mu, \nu)$.

Definizione 8.5 (Distanza di Wasserstein)**.** La *distanza di Wasserstein di ordine* p è:

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Pi(\mu,\nu)} \int_{X \times X} d(x, y)^p \, d\gamma(x, y)\right)^{1/p}. \tag{8.3}$$

Teorema 8.3 (Dualità di Kantorovich-Rubinstein)**.** Per $p = 1$:

$$W_1(\mu, \nu) = \sup_{f : \mathrm{Lip}(f) \le 1} \left[\int f \, d\mu - \int f \, d\nu\right], \tag{8.4}$$

dove il supremo è su tutte le funzioni 1-Lipschitz.

Dimostrazione. La dualità di Fenchel-Moreau applicata al problema lineare di Kantorovich: il duale del problema primale $\inf_\gamma \int c \, d\gamma$ soggetto a $\gamma \in \Pi(\mu, \nu)$ è $\sup_{f,g : f(x)+g(y) \le c(x,y)} [\int f \, d\mu + \int g \, d\nu]$. Per il caso $c(x, y) = d(x, y)$ (distanza), $f(x) + g(y) \le d(x, y)$ implica $f(x) - f(y) \le d(x, y)$ (1-Lipschitz) e $g = f$ (per ottimalità). □

Wasserstein e mean-field games

Nei mean-field games, la misura della popolazione $\mu_t \in \mathcal{P}(\mathbb{R}^d)$ evolve secondo la Fokker-Planck-Kolmogorov (1.26). La distanza di Wasserstein è la metrica naturale per misurare la convergenza dell'apprendimento:

Proposizione 8.1 (Continuità delle MFG in W_2)**.** Le soluzioni (V^*, μ^*) delle MFG dipendono continuamente dai dati iniziali nella metrica W_2: se $\mu_0^{(n)} \to \mu_0$ in W_2, allora le soluzioni corrispondenti convergono $\mu_t^{(n)} \to \mu_t^*$ in W_2 uniformemente su compatti.

Distanza KL e relazione con Wasserstein

Proposizione 8.2 (Disuguaglianza di Talagrand)**.** Per μ assolutamente continua rispetto a ν su $\mathbb{R}^d$:

$$W_2^2(\mu, \nu) \leq 2\,\mathrm{KL}(\mu \| \nu) \cdot \sigma^2, \tag{8.5}$$

dove σ^2 è la varianza di ν (finita per ν sub-gaussiana).

Questa disuguaglianza collega la penalità KL del training RL (come in GRPO) con la distanza W_2 tra la politica corrente e quella di riferimento: minimizzare la KL implica il controllo di W_2.

8.3 Teoria della Misura e Topologia

Il Teorema di Riesz-Markov-Kakutani

Teorema 8.4 (Riesz-Markov-Kakutani)**.** Sia X uno spazio topologico localmente compatto di Hausdorff. Esiste una corrispondenza biunivoca tra:

1. Funzionali lineari positivi $I : C_c(X) \to \mathbb{R}$ (su funzioni continue a supporto compatto);
2. Misure di Borel positive regolari μ su X: $I(f) = \int f \, d\mu$.

Questo teorema è fondamentale per la TDA: il diagramma di persistenza può essere visto come una misura μ_{Dgm} sullo spazio dei punti (b, d):

$$\mu_{\mathrm{Dgm}} = \sum_{(b,d) \in \mathrm{Dgm}} \delta_{(b,d)}, \tag{8.6}$$

dove $\delta_{(b,d)}$ è la misura di Dirac nel punto (b, d). Le statistiche del diagramma (persistence image, landscape) sono funzionali di questa misura.

Categorification dell'omologia persistente

L'omologia persistente ha una formulazione categoriale che chiarisce il Teorema di Struttura (§537). Un *modulo di persistenza* è un funtore $M : (\mathbb{R}, \leq) \to \mathbf{Vec}_{\mathbb{F}}$ dalla categoria dei reali (con morfismi le inclusioni $s \leq t$) alla categoria degli spazi vettoriali su $\mathbb{F}$.

Definizione 8.6 (Decomposizione di un modulo di persistenza)**.** Un modulo di persistenza M è *q-tamable* se ammette una decomposizione in moduli intervallo $\mathbb{F}[b, d)$: esistono insiemi di indici $\{b_j, d_j\}_{j \in J}$ tali che:

$$M \cong \bigoplus_{j \in J} \mathbb{F}[b_j, d_j). \tag{8.7}$$

Teorema 8.5 (Botnan e Crawley-Boevey, 2020)**.** Ogni modulo di persistenza q-tamable su $\mathbb{R}$ si decompone in modo unico come somma diretta di moduli intervallo. Questa unicità è l'analogo del teorema di struttura dei moduli finitamente generati su PID, applicato alla categoria dei moduli di persistenza.

8.4 Geometria dell'Informazione e Reti Neurali

La varietà dei modelli e la metrica di Fisher-Rao

L'insieme dei modelli probabilistici parametrizzati $\{P_\theta : \theta \in \Theta\}$ forma una *varietà statistica* (Amari, 1985). La metrica di Fisher-Rao rende questa varietà una varietà Riemanniana:

$$g_{ij}(\theta) = \mathbb{E}_{P_\theta}\left[\frac{\partial \log P_\theta}{\partial \theta_i} \frac{\partial \log P_\theta}{\partial \theta_j}\right] = F_{ij}(\theta). \tag{8.8}$$

Definizione 8.7 (Connessioni α-duale)**.** La geometria dell'informazione introduce famiglie di connessioni affini parametrizzate da $\alpha \in \mathbb{R}$. Le connessioni $+1$ (esponenziale) e -1 (mixtural) sono duali rispetto alla metrica di Fisher-Rao. Le geodetiche della connessione esponenziale corrispondono alle distribuzioni della famiglia esponenziale.

Proposizione 8.3 (Pitagora statistico)**.** In una varietà statistica, la divergenza KL soddisfa un analogo del teorema di Pitagora: per tre distribuzioni P, Q, R con P sulla geodetica esponenziale tra Q e R:

$$\mathrm{KL}(Q \| R) = \mathrm{KL}(Q \| P) + \mathrm{KL}(P \| R). \tag{8.9}$$

Questa relazione è fondamentale per la derivazione di algoritmi EM e per la comprensione dei metodi di gradiente naturale.

Natural gradient e connessione con Newton

Il gradiente naturale è la direzione di steepest descent nella metrica di Fisher-Rao. Equivalentemente, è la soluzione del problema variazionale:

$$\tilde{\nabla} J(\theta) = \arg\max_{\Delta} \frac{\nabla J(\theta)^\top \Delta}{\sqrt{\Delta^\top F(\theta) \Delta}}, \tag{8.10}$$

che è la direzione che massimizza il guadagno di J per unità di distanza di Fisher-Rao.

Proposizione 8.4 (Natural gradient vs Newton)**.** Il gradiente naturale $F^{-1}\nabla J$ coincide con il passo di Newton quando la loss è il negativo del log-likelihood: $J = -\mathbb{E}[\log P_\theta]$. Per loss generali, il gradiente naturale è un'approssimazione di Newton che usa la geometria della distribuzione invece della curvatura della loss.

Dimostrazione. Il passo di Newton è $-[\nabla^2 J(\theta)]^{-1}\nabla J(\theta)$. Per $J = -\mathbb{E}[\log P_\theta]$: $\nabla^2 J = F(\theta)$ (sotto il vero modello). Quindi il passo di Newton è $-F^{-1}\nabla J$, identico al gradiente naturale. Per loss generali, $\nabla^2 J \neq F$ in generale, ma per politiche softmax la Fisher e la Hessiana della loss di policy gradient coincidono sull'ottimo. □

8.5 Analisi Funzionale e Spazi di Funzioni Neurali

Lo spazio delle funzioni rappresentate da reti neurali

Definizione 8.8 (Spazio di Barron)**.** Lo *spazio di Barron* $\mathcal{B}(\Omega)$ su un dominio compatto $\Omega \subset \mathbb{R}^d$ è lo spazio delle funzioni $f : \Omega \to \mathbb{R}$ tali che:

$$\|f\|_{\mathcal{B}} = \int_{\mathbb{R}^d} |\hat{f}(\omega)| \, \|\omega\|_1 \, d\omega < \infty, \tag{8.11}$$

dove $\hat{f}$ è la trasformata di Fourier di f.

Teorema 8.6 (Barron, 1993)**.** Per ogni $f \in \mathcal{B}(\Omega)$ e ogni $n \geq 1$, esiste una rete neurale a uno strato nascosto con n neuroni f_n tale che:

$$\int_\Omega (f(x) - f_n(x))^2 \, dx \leq \frac{C \, \|f\|_{\mathcal{B}}^2}{n}, \tag{8.12}$$

dove C dipende solo dal diametro di Ω. La dimensione di Ω non compare nell'esponente: le reti neurali evitano la maledizione della dimensionalità per funzioni in

$\mathcal{B}$.

Neural Tangent Kernel e linearizzazione

Per reti neurali con larghezza $m \to \infty$, l'addestramento per discesa del gradiente è equivalente alla regressione kernel con il *Neural Tangent Kernel* (NTK; Jacot et al., 2018):

$$K_{\text{NTK}}(x, x') = \mathbb{E}_{\theta \sim \text{init}}\left[\nabla_\theta f_\theta(x)^\top \nabla_\theta f_\theta(x')\right]. \tag{8.13}$$

Teorema 8.7 (Convergenza NTK)**.** In una rete di larghezza m con inizializzazione casuale gaussiana, la dinamica del training con learning rate η soddisfa:

$$\|f_\theta(t) - f^*_{\text{NTK}}\|_{\mathcal{H}_{K_{\text{NTK}}}} \leq O(m^{-1/2}) \tag{8.14}$$

con alta probabilità, dove f^*_{NTK} è il predittore ottimale nel RKHS del NTK. Il NTK rimane quasi costante durante il training: $K^{(t)}_{\text{NTK}} \approx K^{(0)}_{\text{NTK}}$ con oscillazioni $O(m^{-1/2})$.

Connessione NTK-EWC

La matrice di Fisher nella loss EWC ha una connessione diretta con il NTK:

$$F_{1,jj} = \mathbb{E}_{x \sim \mathcal{D}_1}\left[\left(\frac{\partial f_\theta(x)}{\partial \theta_j}\right)^2\right] = \left[\operatorname{diag}\left(\mathbb{E}_x\left[\nabla_\theta f_\theta(x)\, \nabla_\theta f_\theta(x)^\top\right]\right)\right]_j. \tag{8.15}$$

Quindi la Fisher diagonale di EWC è la diagonale del NTK (calcolata sul dataset del task 1). I parametri con alta Fisher corrispondono alle direzioni in cui la funzione rappresentata dalla rete è più sensibile all'input, che sono esattamente le direzioni più importanti per il task 1.

8.6 Algoritmi di Campionamento per il Test-Time Compute

Sequential Monte Carlo per LLM

Il *Sequential Monte Carlo* (SMC) è un framework generale per campionamento da distribuzioni sequenziali che include best-of-N e MCTS come casi speciali.

dove $\text{ESS} = (\sum_i w_i)^2 / \sum_i w_i^2$ è l'Effective Sample Size, che misura la degenerazione delle particelle.

Proposizione 8.5 (SMC come generalizzazione di best-of-N e MCTS)**.**

- **Best-of-N**: SMC senza resampling e senza aggiornamento intermedio dei pesi (solo punteggio finale).
- **MCTS**: SMC con resampling ad ogni step e politica di selezione UCB1 invece

Algorithm 6 SMC per Generazione con LLM

Require: LLM π_θ, verifier V_ϕ, N particelle, problema x

Inizializza N particelle: $p_i^{(0)} = (x)$, pesi $w_i^{(0)} = 1/N$
for $t = 1, \ldots, T$ **do** ▷ T step di reasoning
  for $i = 1, \ldots, N$ **do**
    Campiona token $r_t^{(i)} \sim \pi_\theta(\cdot | p_i^{(t-1)})$
    Aggiorna particella: $p_i^{(t)} = (p_i^{(t-1)}, r_t^{(i)})$
    Aggiorna peso: $w_i^{(t)} \propto w_i^{(t-1)} \cdot V_\phi(x, p_i^{(t)}) / V_\phi(x, p_i^{(t-1)})$
  end for
  Resampling: se $\mathrm{ESS}(w^{(t)}) < N/2$, ricampiona le particelle con pesi $w^{(t)}$
end forreturn particella $p_i^{(T)}$ con peso massimo

del campionamento proporzionale ai pesi.

- **Beam search**: SMC deterministico (N traiettorie a pesi uguali, resampling = top-N selection).

Analisi della varianza dell'estimatore SMC

La varianza dell'estimatore SMC per $\mathbb{E}_{\pi^*}[R]$ (dove $\pi^* \propto \pi_\theta \cdot V_\phi$) decresce con il numero di particelle:

$$\mathrm{Var}\left[\hat{R}_N^{\mathrm{SMC}}\right] \leq \frac{C_T}{N}, \quad C_T = e^{D_{\chi^2}(\pi^* \| \pi_\theta^{\mathrm{prop}})}, \tag{8.16}$$

dove D_{χ^2} è la χ^2-divergenza tra la distribuzione target π^* e la proposta π_θ. Quando π_θ è vicina a π^* (model già buono), C_T è piccola e la varianza decresce rapidamente con N. Quando π_θ è distante da π^* (problema difficile), C_T è grande e servono molte particelle.

Adaptive Computation Time (ACT)

Un approccio alternativo al test-time compute scaling è l'Adaptive Computation Time (Graves, 2016): la rete decide dinamicamente quanti step computazionali eseguire per ogni input, tramite un meccanismo di halting:

Definizione 8.9 (Halting probability)**.** Al passo t, la rete produce una *halting probability* $h_t \in [0, 1]$. Il numero atteso di step computazionali è:

$$\mathbb{E}[\text{steps}] = \mathbb{E}\left[\min\left\{n : \sum_{t=1}^{n} h_t \geq 1\right\}\right], \tag{8.17}$$

e il costo computazionale è penalizzato proporzionalmente: $\mathcal{L}_{\mathrm{ACT}} = \mathcal{L}_{\mathrm{task}} + \tau\, \mathbb{E}[\text{steps}]$.

L'ACT è concettualmente legato al test-time compute scaling: problemi difficili (alta incertezza) richiedono più step; problemi facili (bassa incertezza) si risolvono rapidamente. La differenza con MCTS è che ACT è un modello sequenziale (un passo alla volta) mentre MCTS è un albero (esplorazione branching).

Capitolo 9

Derivazioni Complete: dalle Equazioni ai Codici

9.1 Derivazione Completa di GRPO da Principi Primi

Formulazione del problema di ottimizzazione vincolata

Partiamo dal problema di ottimizzazione fondamentale per i modelli RL su ragionamento. Dato un modello di riferimento π_{ref} e un insieme di problemi $\mathcal{D} = \{x_1, \ldots, x_n\}$ con verificatore $R(x, y) \in \{0, 1\}$, vogliamo trovare la politica π che massimizza la ricompensa attesa con una penalità KL dal modello di riferimento:

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},\,(T,y)\sim\pi(\cdot|x)}[R(x, y)] \quad \text{s.t.} \quad \mathbb{E}_x \text{KL}(\pi(\cdot|x)\|\pi_{\text{ref}}(\cdot|x)) \leq \delta. \tag{9.1}$$

Soluzione del Lagrangiano

Introducendo il moltiplicatore di Lagrange $\beta > 0$ per il vincolo KL, il Lagrangiano è:

$$\mathcal{L}(\pi, \beta) = \mathbb{E}_{x,(T,y)}[R(x, y)] - \beta\, \mathbb{E}_x \text{KL}(\pi(\cdot|x)\|\pi_{\text{ref}}(\cdot|x)). \tag{9.2}$$

Espandendo la KL e ottimizzando su π con metodo del calcolo delle variazioni:

$$\frac{\delta\mathcal{L}}{\delta\pi(T, y|x)} = R(x, y) - \beta\,(\log \pi(T, y|x) - \log \pi_{\text{ref}}(T, y|x)) - \beta = 0$$

$$\implies \log \pi^*(T, y|x) = \log \pi_{\text{ref}}(T, y|x) + \frac{R(x, y)}{\beta} - 1. \tag{9.3}$$

Normalizzando (la costante -1 diventa parte della funzione di partizione):

$$\pi^*(T, y|x) = \frac{\pi_{\text{ref}}(T, y|x)\, e^{R(x,y)/\beta}}{Z(x, \beta)}, \quad Z(x, \beta) = \mathbb{E}_{(T,y)\sim\pi_{\text{ref}}}\left[e^{R(x,y)/\beta}\right]. \tag{9.4}$$

Questa è la *distribuzione di Gibbs* con temperatura β.

Approssimazione GRPO

La soluzione esatta (9.4) richiede il calcolo della funzione di partizione $Z(x, \beta)$, che è intrattabile. GRPO la approssima campionando G completamenti $\{(T_i, y_i)\}_{i=1}^{G} \sim \pi_{\theta^{\text{old}}}(\cdot|x)$ e stimando il vantaggio normalizzato:

$$\hat{A}_i = \frac{R(x, y_i) - \bar{R}}{\text{std}(R)}, \quad \bar{R} = \frac{1}{G}\sum_{i=1}^{G} R(x, y_i). \tag{9.5}$$

L'interpretazione è che $e^{R_i/\beta} \approx e^{\hat{A}_i \cdot \text{std}(R)/\beta}$: la normalizzazione standardizza le ricompense per adattare la temperatura.

Derivazione della loss surrogate con clipping

La loss GRPO è un surrogate della policy gradient objective con clipping (come PPO, Vol. III, §115). Partendo dall'estimatore IS (importance sampling):

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{x,(T,y)\sim\pi_{\theta^{\text{old}}}}\left[\frac{\pi_\theta(T,y|x)}{\pi_{\theta^{\text{old}}}(T,y|x)}\,\hat{A}\,\nabla_\theta \log \pi_\theta(T,y|x)\right] \\ &= \mathbb{E}\Big[r(\theta)\,\hat{A}\,\nabla_\theta \log \pi_\theta\Big], \quad r(\theta)=\frac{\pi_\theta}{\pi_{\theta^{\text{old}}}}. \end{aligned} \tag{9.6}$$

Il surrogate loss è $L^{\text{CPI}}(\theta) = \mathbb{E}[r(\theta)\hat{A}]$. Il clipping garantisce la stabilità:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}\Big[\min\Big(r(\theta)\hat{A},\ \text{clip}(r(\theta),1-\varepsilon,1+\varepsilon)\hat{A}\Big)\Big]. \tag{9.7}$$

Proposizione 9.1 (GRPO è una lower bound di J)**.** Sia $J_{\text{new}}(\theta) = \mathbb{E}_{\pi_\theta}[R]$ la ricompensa sotto la nuova politica. Allora:

$$J_{\text{new}}(\theta) \geq L^{\text{CLIP}}(\theta) - \frac{2\gamma\varepsilon}{(1-\gamma)^2}\,\mathbb{E}_{s\sim d^{\pi_\theta}}[D_{\text{TV}}(\pi_\theta(\cdot|s),\pi_{\theta^{\text{old}}}(\cdot|s))], \tag{9.8}$$

dove d^{π_θ} è la distribuzione stazionaria degli stati sotto π_θ.

Dimostrazione. Applicando il teorema di policy improvement (Kakade e Langford, 2002):

$$J_{\text{new}} - J_{\text{old}} = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_{\text{new}}}}\mathbb{E}_{a\sim\pi_{\text{new}}}[A^{\pi_{\text{old}}}(s,a)]\,.$$

Sostituendo $d^{\pi_{\text{new}}}$ con $d^{\pi_{\text{old}}}$ si introduce un errore bounded da $\frac{2\gamma}{(1-\gamma)^2}D_{\text{TV}}(\pi_{\text{new}},\pi_{\text{old}})$. Il termine L^{CPI} approssima il lato destro sotto $d^{\pi_{\text{old}}}$. Il clipping introduce un'ulteriore approssimazione, limitata dall'ampiezza ε. □

9.2 Derivazione Completa di EWC: dalla Fisher all'Algoritmo

La matrice di Fisher come curvatura del log-likelihood

Per una famiglia esponenziale $P(y|x,\theta) = h(y)\,e^{\theta^\top T(x,y)-A(\theta)}$, la Fisher coincide con la curvatura del log-likelihood:

$$F(\theta) = -\mathbb{E}_x\mathbb{E}_{y|x}\big[\nabla^2_\theta \log P(y|x,\theta)\big] = \mathbb{E}_x\big[\nabla_\theta \log P\,(\nabla_\theta \log P)^\top\big]\,. \tag{9.9}$$

Per reti neurali con loss di cross-entropy:

$$-\log P(y|x,\theta) = -\sum_c y_c \log \hat{p}_c(x;\theta), \tag{9.10}$$

il gradiente è $g_j = \partial(-\log P)/\partial\theta_j$, e la Fisher diagonale è:

$$F_{jj} = \mathbb{E}_x\mathbb{E}_{y|x}\big[g_j^2\big] = \mathbb{E}_x\left[\sum_c \hat{p}_c\left(\frac{\partial \log \hat{p}_c}{\partial\theta_j}\right)^2\right]. \tag{9.11}$$

Relazione tra Fisher e Hessiana

Per modelli ben specificati (dove $P(y|x,\theta)$ è il vero modello generativo), la matrice di Fisher coincide con la negativa dell'Hessiana del log-likelihood atteso: $F(\theta^*) = -\mathbb{E}[\nabla^2_\theta \log P]|_{\theta^*}$. Per modelli mal specificati (come le reti neurali su dati reali), F e $-\nabla^2\mathcal{L}$ differiscono. La scelta di F invece dell'Hessiana in EWC è motivata da:

1. **Positività semi-definitezza**: $F \geq 0$ sempre, mentre l'Hessiana può avere autovalori negativi (punti di sella).
2. **Costo computazionale**: F si calcola con backpropagation standard; l'Hessiana richiede backpropagation del secondo ordine.
3. **Interpretazione bayesiana**: F^{-1} è la varianza del posteriore gaussiano approssimato, come derivato nella §532.

Algoritmo EWC completo

Algorithm 7 Elastic Weight Consolidation (EWC)

Require: Sequenza di task $\mathcal{T}_1, \mathcal{T}_2, \ldots$, forza $\lambda > 0$, dataset $\mathcal{D}'_k$ per il calcolo della Fisher (subset di $\mathcal{D}_k$)

Addestra su $\mathcal{T}_1$: $\theta^*_1 \leftarrow \arg\min_\theta \mathcal{L}_1(\theta)$
Calcola Fisher diagonale: $F_{1,jj} \leftarrow \frac{1}{|\mathcal{D}'_1|} \sum_{(x,y)\in\mathcal{D}'_1} g_j^2$
for $k = 2, 3, \ldots$ **do**
    Addestra su $\mathcal{T}_k$ con loss EWC: $\theta^*_k \leftarrow \arg\min_\theta \mathcal{L}_k(\theta) + \frac{\lambda}{2}\sum_j F_{k-1,jj}(\theta_j - \theta^*_{k-1,j})^2$
    Calcola Fisher per task corrente: $F_{k,jj} \leftarrow \ldots$
    Aggiorna Fisher accumulata (Online EWC): $\tilde{F}_{k,jj} \leftarrow (1-\alpha)\tilde{F}_{k-1,jj} + \alpha F_{k,jj}$
end for

EWC per Transformer: sfide pratiche

Per un Transformer con $P \sim 10^9$ parametri, l'EWC presenta sfide specifiche:

Sfida 1: Correlazioni tra strati. La Fisher diagonale ignora le correlazioni tra i parametri di strati diversi. Nei Transformer, i layer sono fortemente correlati: il gradiente di un'attenzione head influenza i layer successivi tramite la norma residuale. K-FAC può catturare queste correlazioni a costo ridotto.

Sfida 2: LoRA come EWC implicita. Il fine-tuning con LoRA (Low-Rank Adaptation) addestra solo una perturbazione $\Delta W = BA$ (con $B \in \mathbb{R}^{m\times r}$, $A \in \mathbb{R}^{r\times n}$, $r \ll \min(m,n)$) lasciando i pesi pre-addestrati fissi. Questo è equivalente a EWC con $\lambda \to \infty$ per i pesi della rete base: i pesi importanti per il pre-addestramento non vengono toccati. La differenza è che LoRA non calcola esplicitamente la Fisher, ma la struttura di basso rango di ΔW funge da regolarizzatore implicito.

9.3 Connessioni con la Teoria dei Sistemi Dinamici

Il forgetting come sistema dinamico

L'addestramento sequenziale su task $\mathcal{T}_1, \mathcal{T}_2, \ldots$ può essere visto come un sistema dinamico nello spazio dei parametri $\mathbb{R}^P$:

$$\theta_{t+1} = \theta_t - \eta \, \nabla_\theta \mathcal{L}_k(\theta_t) \quad \text{durante il task } k. \tag{9.12}$$

Questo è il flusso del gradiente discreto. Nella versione continua:

$$\dot{\theta} = -\nabla_\theta \mathcal{L}_k(\theta(t)). \tag{9.13}$$

Definizione 9.1 (Bacino di attrazione di un task)**.** Il *bacino di attrazione* del task k è l'insieme degli stati iniziali che convergono al minimo θ_k^* sotto il flusso del gradiente:

$$B_k = \{\theta_0 \in \mathbb{R}^P : \lim_{t\to\infty} \theta(t) = \theta_k^*\}. \tag{9.14}$$

Il catastrophic forgetting avviene quando $B_1 \cap B_2 = \emptyset$ o quando $\theta_2^* \notin B_1$: il minimo del task 2 è fuori dal bacino del task 1. I metodi di continual learning cercano di trovare un θ che giace nell'intersezione di più bacini.

Proposizione 9.2 (EWC deforma i bacini)**.** La loss EWC $\mathcal{L}_2(\theta) + \frac{\lambda}{2}\Delta\theta^\top F_1 \Delta\theta$ ha il suo minimo θ^*_{EWC} nell'intersezione dei bacini $B_1^{\text{EWC}} \cap B_2$, dove B_1^{EWC} è il bacino modificato da un elissoide di Fisher centrata in θ_1^*. Formalmente:

$$\theta^*_{\text{EWC}} = \arg\min_\theta \mathcal{L}_2(\theta) \quad \text{s.t.} \quad \Delta\theta^\top F_1 \Delta\theta \le c/\lambda. \tag{9.15}$$

Per $\lambda \to \infty$, $\theta^*_{\text{EWC}} \to \theta_1^*$ (nessun apprendimento). Per $\lambda \to 0$, $\theta^*_{\text{EWC}} \to \theta_2^*$ (forgetting totale).

Attractor landscape e multistabilità

Il fenomeno del grokking (Vol. VII, §444) ha un'interpretazione naturale come *multistabilità* nel sistema dinamico del gradiente. Il loss landscape ha due attrattori:

1. **Attractor di memorizzazione** θ^*_{mem}: minimo largo con alta loss di test e bassa loss di training. Alta curvatura locale.
2. **Attractor di generalizzazione** θ^*_{gen}: minimo stretto con bassa loss di test. Bassa curvatura locale.

La discesa del gradiente inizialmente converge a θ^*_{mem} (bacino più grande). Il grokking avviene quando il modello "scappa" da θ^*_{mem} verso θ^*_{gen}, tipicamente per effetto della *weight decay* che restringe il bacino di θ^*_{mem}.

Dalla prospettiva topologica (Sezione E), la transizione corrisponde a:

- Prima: $\beta_0(\mathcal{L}^{-1}(-\infty, c]) = 2$ (due componenti connesse distinte, una per attractor).
- Dopo: $\beta_0 = 1$ (i due bacini si connettono al livello c e il sistema scivola verso θ^*_{gen}).

9.4 Numerical Analysis dei Solver per MCTS

MCTS con Upper Confidence Trees (UCT)

UCT (Kocsis e Szepesvári, 2006) è la versione MCTS standard con UCB1 per la selezione. Il lemma di regret (7.15) si applica con $K = |V|$ (vocabolario), ma il branching factor è trattabile grazie al campionamento:

Teorema 9.1 (UCT con campionamento: regret sull'albero)**.** In MCTS con N_{sim} simulazioni, branching factor campionato b e profondità massima D, la distanza in valore dal nodo ottimale dell'albero è:

$$\left|V^*(x) - V^{\text{MCTS}}(x)\right| \leq \Delta \cdot e^{-N_{\text{sim}}/\tau} + \frac{b^D}{N_{\text{sim}}} O(\ln N_{\text{sim}}), \tag{9.16}$$

dove Δ è il range di ricompense e τ è la costante di tempo di convergenza.

Parallelizzazione di MCTS

Il MCTS per LLM è parallelizzabile su GPU tramite *virtual loss*: si simula la selezione di un nodo come se fosse già stato visitato, permettendo la valutazione parallela di B nodi:

Algorithm 8 MCTS con Virtual Loss Parallelo

Require: Batch size B, modello π_θ, verifier V_ϕ

1: Inizializza l'albero con radice x
2: **repeat**
3: **Selezione parallela**: per $b = 1, \ldots, B$, seleziona v_b con UCB1 e aggiungi virtual loss al percorso
4: **Espansione batch**: genera simultaneamente B step di reasoning con π_θ
5: **Valutazione batch**: calcola $V_\phi(v_b)$ per tutti i b in parallelo su GPU
6: **Backpropagation**: aggiorna q, n e rimuovi virtual loss per ogni b
7: **until** budget esaurito

Il virtual loss previene che più worker selezionino lo stesso nodo, aumentando l'esplorazione. Il throughput scale quasi linearmente con B: $\text{throughput}(B) \approx B \cdot \text{throughput}(1) \cdot (1 - \epsilon_{\text{sync}})$, dove $\epsilon_{\text{sync}} \approx 0.1$–$0.2$ è l'overhead di sincronizzazione.

9.5 Reti di Attenzione e Geometria Riemanniana

Lo spazio delle trasformazioni di attenzione

L'operatore di self-attention (Vol. II, §57) per una sequenza di n token con d_{model} dimensioni è:

$$\text{Attn}(X) = \text{softmax}\left(\frac{XW_Q(XW_K)^\top}{\sqrt{d_k}}\right) XW_V \in \mathbb{R}^{n\times d_v}. \tag{9.17}$$

Lo spazio delle matrici di attenzione $\mathcal{A} = \{A \in [0,1]^{n\times n} : A\mathbf{1} = \mathbf{1}\}$ è il *politopo di Birkhoff* (per matrici doubly stochastic) ristretto alle righe stocastiche. Questo politopo ha una struttura riemanniana naturale come sottovarietà di $\mathbb{R}^{n^2}$.

Proposizione 9.3 (Geodetiche nello spazio delle attenzioni)**.** La geodetica nel politopo di Birkhoff tra due matrici di attenzione A_0, A_1 è:

$$A(t) = \text{softmax}((1-t)\log(A_0) + t\log(A_1))\,, \tag{9.18}$$

dove log e softmax sono applicati riga per riga. Questa geodetica rimane nel politopo per $t \in [0,1]$ per la convessità di softmax(log(·)).

Attenzione come trasporto ottimale

L'operatore di attenzione ha una connessione profonda con il *trasporto ottimale* (Villani, 2003). Il piano di attenzione $A \in [0,1]^{n\times n}$ può essere interpretato come un piano di trasporto tra le posizioni dei query e delle key. La perdita di trasporto è:

$$W_2^2(A) = \sum_{i,j} A_{ij}\, d(i,j)^2, \tag{9.19}$$

dove $d(i,j)$ è la distanza tra posizione i (query) e posizione j (key). Nei modelli linguistici, questa distanza è la distanza posizionale $|i-j|$.

Proposizione 9.4 (Attenzione locale come trasporto ottimale)**.** Minimizzare $W_2^2(A)$ soggetto a $A\mathbf{1} = \mathbf{1}$ e ai logit del modello corrisponde a selezionare l'attenzione *locale*: ogni token attende preferenzialmente i token vicini. Questo spiega perché l'attenzione locale (come in Longformer e BigBird) è un'approssimazione naturale dell'attenzione globale per sequenze lunghe.

Connessione con la TDA: persistenza dell'attenzione

Il grafo di attenzione $G_A = (V, E, w)$ con $w(i,j) = A_{ij}$ ammette una filtrazione naturale: per soglia ε, il sottografo G_A^ε include solo gli archi con peso $A_{ij} \geq \varepsilon$. La persistent homology di questa filtrazione descrive la struttura multi-scala dell'attenzione:

- **Features con alta persistenza**: pattern di attenzione che rimangono stabili al variare di ε — corrispondono a relazioni sintattiche fondamentali (soggetto-verbo, determinante-nome).

- **Features con bassa persistenza**: pattern di attenzione effimeri — corrispondono a relazioni semantiche contestuali.

Studi sperimentali (Clarkson et al., 2023) mostrano che le features ad alta persistenza degli attention head corrispondono a strutture grammaticali universali identificabili con analisi di dipendenze, mentre quelle a bassa persistenza catturano la semantica specifica del contesto.

9.6 Formulazione Variazionale del Continual Learning

Continual Learning come minimizzazione su varietà

Il problema del continual learning può essere riformulato come ottimizzazione su una *varietà vincolata* nello spazio dei parametri. Definiamo:

$$\mathcal{M}_k(\varepsilon) = \left\{\theta \in \mathbb{R}^P : \mathcal{L}_k(\theta) \leq \mathcal{L}_k(\theta_k^*) + \varepsilon\right\} \tag{9.20}$$

come il sottolivello ε-approssimato dell'ottimo del task k. Il continual learning cerca:

$$\theta^* \in \bigcap_{k=1}^{T} \mathcal{M}_k(\varepsilon_k). \tag{9.21}$$

L'intersezione è non vuota solo se i task sono sufficientemente compatibili.

Teorema 9.2 (Condizione di compatibilità)**.** L'intersezione $\bigcap_k \mathcal{M}_k(\varepsilon_k)$ è non vuota se e solo se per ogni coppia (k, k'):

$$d(\theta_k^*, \theta_{k'}^*) \leq r_k(\varepsilon_k) + r_{k'}(\varepsilon_{k'}), \tag{9.22}$$

dove $r_k(\varepsilon) = \max\{\|\theta - \theta_k^*\| : \theta \in \mathcal{M}_k(\varepsilon)\}$ è il "raggio" del sottolivello. Per approssimazione quadratica (EWC): $r_k(\varepsilon) \approx \sqrt{2\varepsilon/\lambda_{\min}(F_k)}$ dove $\lambda_{\min}(F_k)$ è il minimo autovalore della Fisher del task k.

Proiezione alternata per il continual learning

Un algoritmo naturale per trovare $\theta^* \in \bigcap_k \mathcal{M}_k(\varepsilon_k)$ è la *proiezione alternata* (von Neumann, 1950):

$$\theta^{(t+1)} = \Pi_{\mathcal{M}_{k(t)}(\varepsilon_{k(t)})}\left(\theta^{(t)}\right), \tag{9.23}$$

dove $k(t)$ è il task ciclato al passo t e $\Pi_{\mathcal{M}}$ è la proiezione sull'insieme $\mathcal{M}$. Per insiemi convessi, la proiezione alternata converge geometricamente all'intersezione. Per insiemi non convessi (come i sottolivelli delle loss neurali), la convergenza non è garantita, ma la proiezione alternata è euristicamente efficace.

Proposizione 9.5 (GEM come proiezione)**.** L'aggiornamento GEM (4.14) è la proiezione del gradiente g_t sul cono:

$$\mathcal{C} = \{g \in \mathbb{R}^P : \langle g, g_k \rangle \geq 0 \; \forall\, k < t\}. \tag{9.24}$$

Questo cono è la linearizzazione dell'intersezione $\bigcap_{k<t} \mathcal{M}_k(\varepsilon_k)$ intorno a θ_t.

Dimostrazione. La proiezione ortogonale di g_t sul cono $\mathcal{C}$ è esattamente la soluzione del QP (4.14): $\arg\min_{g \in \mathcal{C}} \|g - g_t\|_2^2$. Il cono $\mathcal{C}$ è l'insieme dei gradienti compatibili con tutti i vincoli $\langle g,\, g_k \rangle \geq 0$, che linearizzano la condizione di non-peggioramento $\mathcal{L}_k(\theta - \eta g) \leq \mathcal{L}_k(\theta) + O(\eta^2)$. $\square$

Capitolo 10

Reinforcement Learning Avanzato per Sistemi Agentici

10.1 Reward Shaping e Potential-Based Shaping

Il problema del reward design

Nei sistemi agentici basati su LLM, il segnale di ricompensa $R(x, y) \in \{0, 1\}$ è spesso sparso: si riceve feedback solo alla fine della traiettoria. Il *reward shaping* introduce ricompense intermedie $F(s_t, a_t, s_{t+1})$ per guidare l'apprendimento:

$$R'(s, a, s') = R(s, a, s') + F(s, a, s'). \tag{10.1}$$

Definizione 10.1 (Potential-Based Shaping)**.** Una funzione di shaping F è *potential-based* se esiste una funzione di potenziale $\Phi : S \to \mathbb{R}$ tale che:

$$F(s, a, s') = \gamma\,\Phi(s') - \Phi(s). \tag{10.2}$$

Teorema 10.1 (Ng, Harada e Russell, 1999)**.** Il reward shaping potential-based preserva la politica ottimale: per ogni MDP con ricompensa R e ogni Φ, la politica ottimale sotto $R' = R + F$ coincide con quella ottimale sotto R.

Dimostrazione. Il valore modificato sotto politica π è:

$$\begin{aligned} V'^{\pi}(s) &= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R'(s_t, a_t, s_{t+1}) \mid s_0 = s\right] \\ &= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + \gamma\Phi(s_{t+1}) - \Phi(s_t)\right) \mid s_0\right] \\ &= V^\pi(s) + \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^{t+1}\Phi(s_{t+1}) - \sum_{t=0}^{\infty} \gamma^t \Phi(s_t)\right] \\ &= V^\pi(s) - \Phi(s), \end{aligned} \tag{10.3}$$

dove l'ultima uguaglianza usa il telescoping della serie (convergente per $\gamma < 1$). Poiché $V'^{\pi}(s) = V^\pi(s) - \Phi(s)$ e $\Phi(s)$ non dipende da π, $\arg\max_\pi V'^{\pi}(s) = \arg\max_\pi V^\pi(s)$. □

Process Reward come Shaping

Il PRM $V_\phi^{\mathrm{PRM}}(x, r_{\le t})$ può essere usato come funzione di potenziale $\Phi(s_t) = V_\phi^{\mathrm{PRM}}(x, r_{\le t})$, dando una funzione di shaping:

$$F(s_t, a_t, s_{t+1}) = \gamma\, V_\phi^{\mathrm{PRM}}(x, r_{\le t+1}) - V_\phi^{\mathrm{PRM}}(x, r_{\le t}). \tag{10.4}$$

Per il teorema di Ng et al., questo shaping preserva la politica ottimale e accelera l'apprendimento fornendo segnali di ricompensa densi (a ogni step di ragionamento).

10.2 Teoria degli Algoritmi Evolutivi per MARL

Evolutionary Game Theory e Dinamiche Replicative

Definizione 10.2 (Dinamica replicativa)**.** In un gioco simmetrico a N strategie con matrice di payoff A, la *dinamica replicativa* descrive l'evoluzione della distribuzione $x \in \Delta^{N-1}$ delle strategie nella popolazione:

$$\dot{x}_i = x_i\big[(Ax)_i - x^\top Ax\big], \quad i = 1, \dots, N. \tag{10.5}$$

Il termine $(Ax)_i$ è il payoff atteso della strategia i contro la popolazione x, e $x^\top Ax$ è il payoff medio.

Teorema 10.2 (Nash come punti fissi della dinamica replicativa)**.** Ogni equilibrio di Nash x^* del gioco simmetrico è un punto fisso della dinamica replicativa. I punti fissi interni (dove $x_i > 0$ per ogni i) sono tutti equilibri di Nash.

Dimostrazione. In un punto fisso interno: $\dot{x}_i = 0$ implica $(Ax)_i = x^\top Ax$ per ogni i con $x_i > 0$. Questo significa che tutte le strategie nella strategia mista ottengono lo stesso payoff, che è la condizione di equilibrio di Nash per politiche miste. □

Connessione con il MARL

La dinamica replicativa è il continuo analogico della Q-learning dynamics nel MARL: se ogni agente aggiorna la propria politica proporzionalmente alla differenza tra il valore dell'azione corrente e il valore medio, si ottiene un'approssimazione discreta della dinamica replicativa:

$$\pi_{t+1}^i(a) = \frac{\pi_t^i(a)\, e^{\eta Q_t^i(s,a)}}{\sum_{a'} \pi_t^i(a')\, e^{\eta Q_t^i(s,a')}}. \tag{10.6}$$

Questo è l'algoritmo *Multiplicative Weights Update* (MWU), equivalente alla dinamica replicativa per $\eta \to 0$.

Proposizione 10.1 (Convergenza MWU agli equilibri correlati)**.** L'algoritmo MWU con learning rate $\eta_t = \sqrt{\ln N/t}$ converge agli equilibri correlati in media:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\max_{a}\sum_{a'}\hat{\sigma}_t(a,a')\left[R(a,\cdot)-R(a_t,\cdot)\right]\right] \leq O\left(\sqrt{\frac{\ln N}{T}}\right), \tag{10.7}$$

dove $\hat{\sigma}_t$ è la distribuzione empirica congiunta fino al tempo t.

10.3 Teoria dei Grafi Computazionali per l'Addestramento

Il grafo computazionale come DAG

L'addestramento di reti neurali tramite backpropagation può essere formalizzato come elaborazione su un *Directed Acyclic Graph* (DAG) computazionale.

Definizione 10.3 (Grafo computazionale)**.** Un *grafo computazionale* $G = (V, E, f)$ è un DAG dove:

- $V = V_{\text{in}} \cup V_{\text{op}} \cup V_{\text{out}}$ è l'insieme dei nodi (input, operazioni, output);
- $E \subseteq V \times V$ sono gli archi (dipendenze);
- $f : V_{\text{op}} \to \mathcal{F}$ assegna a ogni nodo operazione una funzione differenziabile.

Il forward pass calcola il valore di ogni nodo in ordine topologico. Il backward pass (backpropagation) calcola i gradienti tramite la regola della catena applicata in ordine topologico inverso:

$$\frac{\partial L}{\partial v_j} = \sum_{k:(j,k)\in E} \frac{\partial L}{\partial v_k}\frac{\partial v_k}{\partial v_j}. \tag{10.8}$$

Complexity del backward pass

Proposizione 10.2 (Complessità del backward pass)**.** Per un grafo computazionale con $|V_{\text{op}}|$ operazioni e $|E|$ archi, il backward pass ha complessità $O(|E|)$ (uguale al forward pass) e richiede la memorizzazione di tutti i valori intermedi: $O(|V_{\text{op}}|)$ in memoria.

Per le reti neurali con miliardi di parametri, questa memoria è proibitiva. Le tecniche di *gradient checkpointing* riducono la memoria a $O(\sqrt{|V|})$ eseguendo il forward pass parzialmente durante il backward:

- Si salvano solo i valori ai nodi di checkpoint (ogni $\sqrt{L}$ layer).
- Durante il backward, si ricalcola il forward dal checkpoint più vicino.
- Trade-off: $O(\sqrt{L})$ memoria vs $O(\sqrt{L})$ forward pass extra.

Automatic Differentiation e il caso del MARL

Nel MARL con MAPPO, il backward pass deve propagare i gradienti attraverso le politiche di tutti gli agenti. Per N agenti con politiche π_{θ_i}, il grafo computazionale ha N sottografi separati (uno per agente) connessi solo attraverso il valore congiunto Q_{tot}:

$$\frac{\partial \mathcal{L}_{\text{MAPPO}}}{\partial \theta_i} = \frac{\partial \mathcal{L}_{\text{MAPPO}}}{\partial Q_{\text{tot}}} \cdot \frac{\partial Q_{\text{tot}}}{\partial Q_i} \cdot \frac{\partial Q_i}{\partial \theta_i}. \tag{10.9}$$

La fattorizzazione QMIX garantisce che $\partial Q_{\text{tot}}/\partial Q_i \geq 0$ (IGM), e il backward pass è quindi sempre ben definito (nessun gradiente esplosivo dalla non-monotonicità).

10.4 Analisi di Stabilità dei Video Diffusion Models

Stabilità della generazione video: il problema del flickering

Un problema pratico dei Video Diffusion Model è il *flickering*: variazioni temporalmente incoerenti tra frame consecutivi. Formalmente, il flickering si misura con:

$$\text{Flicker}(v) = \frac{1}{T-1} \sum_{t=1}^{T-1} \|v_{t+1} - v_t\|_2^2 \,. \tag{10.10}$$

Proposizione 10.3 (Flickering come varianza temporale)**.** In un Video Diffusion Model con attenzione temporale fattorizzata, il flickering atteso è:

$$\mathbb{E}[\text{Flicker}(v)] = 2\,\text{Var}(v_t) - 2\,\text{Cov}(v_t, v_{t+1}), \tag{10.11}$$

dove la varianza e la covarianza sono calcolate sulla distribuzione generativa del modello. Il flickering è zero se $v_t = v_{t+1}$ quasi certamente (video statico) e massimo se i frame sono indipendenti (nessuna coerenza temporale).

Dimostrazione. $\mathbb{E}[\|v_{t+1} - v_t\|_2^2] = \mathbb{E}[\|v_{t+1}\|_2^2] - 2\mathbb{E}[v_{t+1}^\top v_t] + \mathbb{E}[\|v_t\|_2^2] = \text{Var}(v_{t+1}) + \mu_{t+1}^2 - 2\langle \mu_{t+1}, \mu_t \rangle - 2\text{Cov}(v_{t+1}, v_t) + \text{Var}(v_t) + \mu_t^2$. Per video stazionari ($\mu_t = \mu$ e $\text{Var}(v_t) = \sigma^2$ costanti): $= 2\sigma^2 - 2\text{Cov}(v_t, v_{t+1})$. □

Temporal self-attention come regolarizzatore

L'attenzione temporale (Sezione C, §526) agisce come un regolarizzatore che aumenta $\text{Cov}(v_t, v_{t+1})$, riducendo il flickering. Formalmente, per un'attenzione temporale con kernel gaussiano:

$$\text{TemporalAttn}(z)_t = \sum_{t'} \alpha_{t,t'}\, z_{t'}, \quad \alpha_{t,t'} \propto e^{-|t-t'|^2/(2\sigma_{\text{temp}}^2)}, \tag{10.12}$$

la covarianza temporale delle rappresentazioni latenti è amplificata:

$$\text{Cov}(\text{TemporalAttn}(z)_t, \text{TemporalAttn}(z)_{t+1}) \geq \text{Cov}(z_t, z_{t+1}) + O(e^{-1/(2\sigma_{\text{temp}}^2)}). \tag{10.13}$$

Un σ_{temp} grande produce video più fluidi (ma potenzialmente sfocati); un σ_{temp} piccolo produce video più nitidi (ma con più flickering).

10.5 Teoria dell'Informazione Topologica

Entropia dei diagrammi di persistenza

Definizione 10.4 (Entropia di persistenza)**.** Dato un diagramma di persistenza Dgm con persistenze $\ell_k = d_k - b_k$, l'*entropia di persistenza* è:

$$E(\mathrm{Dgm}) = -\sum_k p_k \log p_k, \quad p_k = \frac{\ell_k}{\sum_{k'} \ell_{k'}}. \tag{10.14}$$

Proposizione 10.4 (Proprietà dell'entropia di persistenza)**.** L'entropia di persistenza E soddisfa:

1. $E = 0$ se e solo se c'è un'unica feature topologica (diagramma con un solo punto).
2. E è massima (log del numero di features) se tutte le features hanno la stessa persistenza.
3. E è continua rispetto alla distanza bottleneck: $|E(A) - E(B)| \leq C\, d_\infty(A, B)^{1/2}$.

Dimostrazione. [Sketch della proprietà 3] La distanza bottleneck $d_\infty(A, B) \leq \varepsilon$ implica che ogni coppia di features corrispondenti $(b_k, d_k) \leftrightarrow (b'_k, d'_k)$ soddisfa $|(d_k - b_k) - (d'_k - b'_k)| \leq 2\varepsilon$. Quindi la variazione delle probabilità $|p_k - p'_k| \leq 4\varepsilon / \sum_k \ell_k$. La variazione dell'entropia segue dalla Lipschitz-continuità di $-p \log p$. □

Mutual information topologica

Definizione 10.5 (Mutual information topologica tra layer)**.** Per due layer l e l' di una rete neurale con rappresentazioni $H^{(l)}, H^{(l')} \in \mathbb{R}^{n \times d}$, la *mutual information topologica* è:

$$I_{\mathrm{top}}(l, l') = E(\mathrm{Dgm}(H^{(l)})) + E(\mathrm{Dgm}(H^{(l')})) - E(\mathrm{Dgm}([H^{(l)}, H^{(l')}])), \tag{10.15}$$

dove $[H^{(l)}, H^{(l')}]$ è la concatenazione e il diagramma è calcolato nello spazio congiunto $\mathbb{R}^{2d}$.

Questa quantità misura la ridondanza topologica tra i due layer: se le strutture topologiche di $H^{(l)}$ e $H^{(l')}$ sono identiche, $I_{\mathrm{top}} = E(\mathrm{Dgm}(H^{(l)}))$. Se sono indipendenti, $I_{\mathrm{top}} = 0$.

Betti numbers e codifica dell'informazione

Proposizione 10.5 (Bound di capacità tramite Betti numbers)**.** Per un classificatore binario $f_\theta : \mathbb{R}^d \to \{0, 1\}$ con decision boundary $\{x : f_\theta(x) = 0.5\}$, il numero di componenti connesse della regione di decisione $\{x : f_\theta(x) \geq 0.5\}$ soddisfa:

$$\beta_0(\{x : f_\theta(x) \geq 0.5\}) \leq O(W^L), \tag{10.16}$$

dove W è la larghezza massima dei layer e L è il numero di layer. Per reti con attivazioni ReLU, questo bound è stretto (Montufar et al., 2014).

Questo risultato collega la topologia della funzione appresa con la capacità espressiva della rete: reti più profonde e larghe possono rappresentare funzioni con più componenti connesse, permettendo di separare distribuzioni topologicamente complesse.

10.6 Algoritmi di Ottimizzazione di Secondo Ordine

Newton-CG per il Continual Learning

L'algoritmo di Gradiente Coniugato (CG) permette di calcolare il prodotto $H^{-1}g$ (dove H è l'Hessiana e g il gradiente) in tempo $O(P)$ per iterazione, senza memorizzare H esplicitamente. Questo è cruciale per EWC con Fisher completa:

Algorithm 9 Hessian-Free Optimization per EWC

Require: Loss $\mathcal{L}_2$, Fisher F_1, parametri correnti θ, iterazioni CG K

Calcola gradiente $g = \nabla_\theta(\mathcal{L}_2(\theta) + \frac{\lambda}{2}\|\theta\|_F^2)$

Inizializza $r_0 = g$, $p_0 = r_0$, $x_0 = 0$

for $k = 0, \ldots, K-1$ **do**

  Calcola il prodotto Hessian-vector senza materializzare la Hessiana:

$$Hp_k = \nabla_\theta\left[(\nabla_\theta \mathcal{L}_2(\theta))^\top p_k\right] + \lambda F_1 p_k$$

  $\alpha_k = r_k^\top r_k / (p_k^\top H p_k)$

  $x_{k+1} = x_k + \alpha_k p_k$, $r_{k+1} = r_k - \alpha_k H p_k$

  $\beta_k = r_{k+1}^\top r_{k+1} / r_k^\top r_k$

  $p_{k+1} = r_{k+1} + \beta_k p_k$

end forreturn $\theta_{\text{new}} = \theta - x_K$ (passo di Newton approssimato)

Il prodotto Hessian-vector $Hp_k = \nabla_\theta[(\nabla_\theta \mathcal{L}_2)^\top p_k]$ si calcola con due backward pass in $O(P)$ (R-operator, Pearlmutter 1994). Il termine $\lambda F_1 p_k$ si calcola con una singola passata sui dati del task 1.

L-BFGS per loss landscape nel continual learning

Il metodo L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) approssima $F^{-1}g$ con m coppie (s_k, y_k) di differenze di parametri e gradienti:

$$s_k = \theta_{k+1} - \theta_k, \quad y_k = g_{k+1} - g_k, \quad \rho_k = 1/(y_k^\top s_k). \tag{10.17}$$

L'approssimazione di H^{-1} è:

$$H_k^{-1} = (I - \rho_{k-1} s_{k-1} y_{k-1}^\top)\, H_{k-1}^{-1}\, (I - \rho_{k-1} y_{k-1} s_{k-1}^\top) + \rho_{k-1} s_{k-1} s_{k-1}^\top. \tag{10.18}$$

La ricorsione two-loop di L-BFGS calcola $H_k^{-1} g$ in $O(mP)$ tempo e $O(mP)$ memoria (contro $O(P^2)$ di BFGS completo).

Per il continual learning, la memoria L-BFGS delle coppie (s_k, y_k) funge da proxy della curvatura del loss landscape: le coppie corrispondenti a task precedenti forniscono informazione implicita sulla geometria di quei task, analogamente a EWC.

10.7 Analisi degli Errori nelle Pipeline di Generazione Video

Error propagation nel 3D-VAE

La loss di ricostruzione (3.2) del 3D-VAE introduce un errore di quantizzazione che si propaga al modello di diffusione. Per una loss di ricostruzione $\mathcal{L}_{\text{rec}} = \mathbb{E}[\|v - \hat{v}\|_2^2] \leq \varepsilon_{\text{VAE}}$, l'errore nel diagramma di persistenza delle attivazioni del decoder è bounded:

$$d_\infty(\text{Dgm}(v), \text{Dgm}(\hat{v})) \leq L_D\, \varepsilon_{\text{VAE}}^{1/2}, \tag{10.19}$$

dove L_D è la costante di Lipschitz del decoder D_ϕ. Questo bound segue dalla stabilità del diagramma di persistenza (§538) e dalla Lipschitz-continuità del decoder.

Cascata di errori nel pipeline

Il pipeline completo di generazione video ha tre stadi:

1. Encoder del 3D-VAE: $v \mapsto z$ con errore ε_{enc}.
2. Diffusion model: $z \mapsto \hat{z}$ con errore $\varepsilon_{\text{diff}}$.
3. Decoder del 3D-VAE: $\hat{z} \mapsto \hat{v}$ con errore ε_{dec}.

Per la disuguaglianza triangolare, l'errore totale è:

$$\|v - \hat{v}\|_2 \leq \|v - D_\phi(z)\|_2 + L_D\, \|z - \hat{z}\|_2 \leq \varepsilon_{\text{dec}} + L_D\, (\varepsilon_{\text{enc}}^{1/2} + \varepsilon_{\text{diff}}). \tag{10.20}$$

Proposizione 10.6 (Bottleneck della pipeline)**.** Sia C_{tot} il compute totale della pipeline. Il trade-off ottimale tra VAE e diffusion model minimizza l'errore totale soggetto a: $C_{\text{VAE}} + C_{\text{diff}} = C_{\text{tot}}$. Con scaling $\varepsilon_{\text{VAE}} \propto C_{\text{VAE}}^{-\alpha_V}$ e $\varepsilon_{\text{diff}} \propto C_{\text{diff}}^{-\alpha_D}$, il compute ottimale per la VAE è:

$$C_{\text{VAE}}^* = \frac{\alpha_V}{\alpha_V + \alpha_D/2}\, C_{\text{tot}}. \tag{10.21}$$

Per $\alpha_V \approx \alpha_D$ (stesso scaling), si investe circa il 66% del compute nel diffusion model e il 34% nella VAE.

Esercizio 5: EWC e Fisher Information su un modello gaussiano

Problema. Considera un modello a due parametri $\theta = (\theta_1, \theta_2)$ con log-likelihood del task 1:

$$\log P(\mathcal{D}_1|\theta) = -\frac{n_1}{2}\left[(\theta_1-\mu_1)^2 + (\theta_2-\mu_2)^2\right].$$

Il task 2 ha loss $\mathcal{L}_2(\theta) = (\theta_1+\theta_2-c)^2/2$ per una costante c. Calcola: (a) la Fisher diagonale di task 1; (b) il minimo EWC per task 2; (c) il forgetting score.

Soluzione.

(a) Fisher diagonale. Dal log-likelihood:

$$\frac{\partial \log P}{\partial \theta_1} = -n_1(\theta_1-\mu_1), \quad \frac{\partial \log P}{\partial \theta_2} = -n_1(\theta_2-\mu_2).$$

Quindi $F_{1,11} = n_1$, $F_{1,22} = n_1$ (costanti, indipendenti da θ).

(b) Minimo EWC. La loss EWC è:

$$\mathcal{L}_{\text{EWC}}(\theta) = \frac{(\theta_1+\theta_2-c)^2}{2} + \frac{\lambda n_1}{2}\left[(\theta_1-\mu_1)^2 + (\theta_2-\mu_2)^2\right].$$

Le condizioni del primo ordine:

$$\begin{aligned}(\theta_1+\theta_2-c) + \lambda n_1(\theta_1-\mu_1) &= 0,\\ (\theta_1+\theta_2-c) + \lambda n_1(\theta_2-\mu_2) &= 0.\end{aligned}$$

Sottraendo: $\lambda n_1(\theta_1-\mu_1) = \lambda n_1(\theta_2-\mu_2)$, da cui $\theta_1-\theta_2 = \mu_1-\mu_2$. Sommando e sostituendo:

$$\theta_1^{\text{EWC}} = \frac{\lambda n_1\mu_1 + c - (\mu_1-\mu_2)/2}{1+\lambda n_1} + \frac{\mu_1-\mu_2}{2}, \quad \theta_2^{\text{EWC}} = \theta_1^{\text{EWC}} - (\mu_1-\mu_2).$$

(c) Forgetting score. Il forgetting è:

$$\mathcal{L}_1(\theta^*_{\text{EWC}}) - \mathcal{L}_1(\theta^*_1) = \frac{n_1}{2}\left[(\theta_1^{\text{EWC}}-\mu_1)^2 + (\theta_2^{\text{EWC}}-\mu_2)^2\right].$$

Per $\lambda \to \infty$: $\theta^*_{\text{EWC}} \to (\mu_1,\mu_2) = \theta^*_1$, forgetting $\to 0$. Per $\lambda \to 0$: $\theta^*_{\text{EWC}} \to \theta^*_2$, forgetting massimo. □

Esercizio 6: Diagramma di persistenza e stabilità

Problema. Considera due nuvole di punti: $\mathcal{X} = \{(0,0),(2,0),(1,\sqrt{3})\}$ (triangolo equilaterale di lato 2) e $\mathcal{X}' = \{(0,0),(2,0),(1,\sqrt{3}+\varepsilon)\}$ (triangolo perturbato). Calcola $\text{Dgm}_1(\mathcal{X})$ e $\text{Dgm}_1(\mathcal{X}')$ e verifica il teorema di stabilità.

Soluzione. Per $\mathcal{X}$ (triangolo equilaterale, lato = 2):

- $\varepsilon = 0$: 3 vertici isolati, $\beta_0 = 3$.
- $\varepsilon = 2$: i 3 lati compaiono simultaneamente (tutti di lunghezza 2), $\beta_0 = 1$, nasce $\beta_1 = 1$ (il ciclo del triangolo).

- $\varepsilon = 2$: il triangolo (2-simplesso) chiude il ciclo — ma in VR, il triangolo compare quando $\varepsilon \geq \max(\text{lati}) = 2$. Quindi il ciclo nasce e muore a $\varepsilon = 2$: degenerato.

In realtà il ciclo nasce quando compaiono tutti e 3 gli spigoli ($\varepsilon = 2$) e muore quando compare il triangolo ($\varepsilon = 2$): persistenza $= 0$. Il diagramma $\mathrm{Dgm}_1(\mathcal{X})$ ha un solo punto degenere sulla diagonale.

Per $\mathcal{X}'$ con $\varepsilon_{\text{pert}} > 0$ piccolo: le distanze cambiano di $O(\varepsilon_{\text{pert}})$. Per il teorema di stabilità:

$$d_\infty(\mathrm{Dgm}_1(\mathcal{X}), \mathrm{Dgm}_1(\mathcal{X}')) \leq \|f_{\mathcal{X}} - f_{\mathcal{X}'}\|_\infty = O(\varepsilon_{\text{pert}}),$$

dove $f_{\mathcal{X}}$ è la funzione di filtrazione (distanza massima in ogni simplesso). La perturbazione di $O(\varepsilon_{\text{pert}})$ nelle coordinate si traduce in una perturbazione di $O(\varepsilon_{\text{pert}})$ nella filtrazione e quindi nel diagramma: la stabilità è verificata. □

10.8 Connessioni con la Fisica Statistica

Reti neurali come sistemi spin-glass

La connessione tra reti neurali e sistemi spin-glass (Mézard, Parisi e Virasoro, 1987) è una delle più profonde della fisica teorica applicata al machine learning. Un sistema spin-glass è un sistema fisico con interazioni disordinate tra variabili binarie $\sigma_i \in \{-1, +1\}$:

$$H(\sigma) = -\sum_{ij} J_{ij}\, \sigma_i \sigma_j - h \sum_i \sigma_i, \tag{10.22}$$

dove J_{ij} sono coppie di interazione casuali e h è un campo esterno.

Proposizione 10.7 (Analogia spin-glass / rete neurale)**.** Una rete neurale con pesi W_{ij} e attivazioni binarie $\sigma_i \in \{-1, +1\}$ implementa esattamente un sistema spin-glass con $J_{ij} = W_{ij}/2$. La fase di addestramento corrisponde alla ricerca del ground state di $H(\sigma; J, h)$ dove i coupling J_{ij} sono i pesi e il campo h è il bias.

Metodo delle repliche e calcolo della loss di equilibrio

Il metodo delle repliche (Parisi, 1979) calcola la free energy di sistemi disordinati tramite il trucco $\log Z = \lim_{n\to 0}(Z^n - 1)/n$:

$$F = -\lim_{n\to 0} \frac{1}{n} \log \mathbb{E}_J[Z^n], \quad Z = \sum_\sigma e^{-\beta H(\sigma; J)}. \tag{10.23}$$

Z^n corrisponde a n copie (repliche) del sistema con gli stessi coupling J:

$$\mathbb{E}_J[Z^n] = \sum_{\sigma^1, \ldots, \sigma^n} \mathbb{E}_J\left[e^{-\beta \sum_\alpha H(\sigma^\alpha)}\right]. \tag{10.24}$$

La media su J (averaging del disordine) introduce overlap tra repliche: $q^{\alpha\beta} = \frac{1}{N}\sum_i \sigma_i^\alpha \sigma_i^\beta$.

Definizione 10.6 (Replica Symmetry Breaking)**.** La soluzione *replica-simmetrica* (RS) assume $q^{\alpha\beta} = q$ costante per $\alpha \neq \beta$. La soluzione *con rottura della simmetria di replica* (RSB) ammette $q^{\alpha\beta}$ dipendente dalla coppia: indica la presenza di molti stati metastabili (ottimi locali multipli).

Per le reti neurali overparametrizzate, la struttura RS è coerente con il fatto che il loss landscape ha un'unica componente connessa di minimi ($\beta_0 = 1$ per la filtrazione di livello, come discusso nella §539). La rottura della simmetria di replica corrisponde topologicamente a $\beta_0 > 1$: minimi disconnessi.

Temperature e sharp threshold nella generalizzazione

La transizione di fase nella generalizzazione (incluso il grokking) può essere analizzata tramite la temperatura statistica $T_s = 1/\beta_s$ (inverso della forza di regolarizzazione):

$$P_{\text{generalizzazione}} = \frac{1}{1 + e^{-\Delta F / T_s}}, \tag{10.25}$$

dove $\Delta F = F_{\text{mem}} - F_{\text{gen}}$ è la differenza di free energy tra la fase di memorizzazione e quella di generalizzazione. La transizione è sharp (discontinua nel limite termodinamico $N \to \infty$) se e solo se $\Delta F = 0$ ha una soluzione unica in T_s.

Proposizione 10.8 (Sharp threshold per il grokking)**.** In un modello di campo medio per la generalizzazione, il grokking avviene a una temperatura critica $T_s^* = \Delta F / \Delta S$, dove ΔS è la differenza di entropia tra le due fasi. Per $T_s < T_s^*$ (forte regolarizzazione), il modello generalizza; per $T_s > T_s^*$ (debole regolarizzazione), memorizza.

Capitolo 11

Teoria Avanzata dell'Ottimizzazione per il Deep Learning

11.1 Loss Landscape: Geometria e Ottimizzazione

Struttura dei punti critici nelle reti overparametrizzate

Definizione 11.1 (Indice di Morse)**.** Per una funzione di Morse $\mathcal{L} : \mathbb{R}^P \to \mathbb{R}$, l'*indice di Morse* di un punto critico θ^* è il numero di autovalori negativi della matrice Hessiana $\nabla^2\mathcal{L}(\theta^*)$. I punti critici di indice 0 sono minimi locali; quelli di indice 1 sono punti di sella di primo ordine; quelli di indice k sono punti di sella di ordine k.

Teorema 11.1 (Dauphin et al., 2014 — Dominanza delle selle nelle reti profonde)**.** Per reti neurali profonde con pesi casuali, la probabilità che un punto critico a loss $\mathcal{L}^*$ sia un minimo locale (indice 0) decade esponenzialmente con la dimensione P:

$$P(\text{indice} = 0 \mid \mathcal{L}(\theta^*) = \mathcal{L}^*) \propto e^{-cP}, \tag{11.1}$$

mentre i punti di sella di ordine $\Theta(P)$ sono dominanti. Tuttavia, tutti i punti critici a loss simile al minimo globale hanno loss quasi uguale (no poor local minima hypothesis), giustificando l'ottimizzazione pratica con SGD.

Dimostrazione. [Sketch] Si modella la Hessiana $H = \nabla^2\mathcal{L}$ come una matrice casuale del *Gaussian Orthogonal Ensemble* (GOE). Per il GOE di dimensione P, la distribuzione degli autovalori converge alla legge di Wigner (semi-cerchio) con raggio $2\sqrt{P}$. La probabilità che tutti gli autovalori siano positivi (minimo) è la probabilità che il minimo autovalore del GOE sia positivo, che scala come e^{-cP} per la legge degli autovalori estremi del GOE. □

Sharpness e generalizzazione: Flat Minima

Definizione 11.2 (Sharpness di un minimo)**.** La *sharpness* di un minimo θ^* con perturbazione $\rho > 0$ è:

$$\text{sharp}_\rho(\theta^*) = \max_{\|\epsilon\|_2 \leq \rho} \mathcal{L}(\theta^* + \epsilon) - \mathcal{L}(\theta^*). \tag{11.2}$$

Un minimo è *flat* se $\text{sharp}_\rho(\theta^*)$ è piccolo per ρ grande; è *sharp* se la loss cresce rapidamente attorno a θ^*.

Proposizione 11.1 (Flat Minima e generalizzazione (Hochreiter e Schmidhuber, 1997))**.** Sia θ^* un minimo della loss di training con sharpness S. Il bound di generalizzazione PAC-Bayes è:

$$\mathcal{L}_{\text{test}}(\theta^*) \leq \mathcal{L}_{\text{train}}(\theta^*) + O\left(\sqrt{\frac{S \cdot P \cdot \log(1/\delta)}{n}}\right), \tag{11.3}$$

dove n è il numero di training samples e δ è la confidenza. I minimi flat (bassa S) generalizzano meglio.

SAM: Sharpness-Aware Minimization

SAM (Foret et al., 2021) minimizza esplicitamente la sharpness durante il training:

$$\min_\theta \max_{\|\epsilon\|_2 \leq \rho} \mathcal{L}(\theta + \epsilon). \tag{11.4}$$

Il passo SAM si calcola in due fasi:

1. **Perturbazione**: $\hat{\epsilon} = \rho \, \nabla_\theta \mathcal{L}(\theta) / \left\|\nabla_\theta \mathcal{L}(\theta)\right\|_2$
2. **Gradiente SAM**: $g_{\text{SAM}} = \nabla_\theta \mathcal{L}(\theta + \hat{\epsilon})$

Proposizione 11.2 (SAM converge a minimi più flat)**.** Sotto ipotesi di regolarità, i minimi raggiunti da SAM soddisfano:

$$\lambda_{\max}(\nabla^2 \mathcal{L}(\theta^*_{\text{SAM}})) \leq \lambda_{\max}(\nabla^2 \mathcal{L}(\theta^*_{\text{SGD}})) \cdot (1 + O(\rho)), \tag{11.5}$$

ovvero SAM tende a minimi con autovalore massimo dell'Hessiana più piccolo, corrispondente a regioni più flat del loss landscape.

Connessione tra SAM e il continual learning

SAM è naturalmente connesso al continual learning: un minimo flat di task 1 è più robusto al forgetting quando si addestra su task 2, perché la loss di task 1 varia poco nelle direzioni di alta sharpness. Formalmente, dall'approssimazione quadratica (4.3):

$$\mathcal{L}_1(\theta_2^*) - \mathcal{L}_1(\theta_1^*) \approx \frac{1}{2}(\theta_2^* - \theta_1^*)^\top H_1 (\theta_2^* - \theta_1^*) \leq \frac{\lambda_{\max}(H_1)}{2} \left\|\theta_2^* - \theta_1^*\right\|^2. \tag{11.6}$$

Per minimi SAM, $\lambda_{\max}(H_1)$ è piccolo, quindi il forgetting è limitato anche per spostamenti $\theta_2^* - \theta_1^*$ grandi. Questa osservazione ha portato a *SAM-EWC*: un metodo che

combina SAM (per trovare minimi flat) con EWC (per la regolarizzazione sui task precedenti).

11.2 Convergenza degli Algoritmi di Generazione: Analisi Quantitativa

DDPM: tasso di convergenza verso la distribuzione dei dati

Teorema 11.2 (De Bortoli et al., 2021 — Convergenza quantitativa dei score-based models)**.** Sia p_{data} la distribuzione dei dati e p_θ^T la distribuzione generata dal DDPM dopo T passi di denoising. Sotto ipotesi di regolarità (score Lipschitz, seconda derivata bounded), il TV distance è:

$$\text{TV}(p_\theta^T, p_{\text{data}}) \leq O\left(\frac{L^2}{T} + \varepsilon_{\text{score}}\sqrt{T}\right), \tag{11.7}$$

dove L è la costante di Lipschitz dello score e $\varepsilon_{\text{score}}$ è l'errore di stima dello score.

Il bound è minimizzato per $T^* = O(L/\varepsilon_{\text{score}})^{2/3}$, dando un errore ottimale di $O((L\,\varepsilon_{\text{score}})^{2/3})$. Questo fornisce una legge di scaling per il numero di passi di denoising in funzione della qualità del modello di score.

Consistency Models: garanzie di one-step generation

Proposizione 11.3 (Errore di generazione in un passo)**.** Per un Consistency Model con errore di training $\mathcal{L}_{\text{CM}} \leq \varepsilon_{\text{CM}}$ (sulla metrica LPIPS), l'errore della generazione in un passo è:

$$\mathbb{E}_{v\sim p_\theta}[\text{LPIPS}(v, v^*)] \leq \varepsilon_{\text{CM}} + O\left(\frac{T\,\varepsilon_{\text{ODE}}}{K}\right), \tag{11.8}$$

dove ε_{ODE} è l'errore dell'ODE solver usato per costruire il target nel training e K è il numero di passi dell'ODE solver.

Flow Matching: convergenza e sample quality

Per il flow matching, la loss di training (7.15) è direttamente un upper bound sull'errore di generazione:

Proposizione 11.4 (Flow Matching è uno score matching reparametrizzato)**.** La loss di flow matching $\mathcal{L}_{\text{FM}}$ è equivalente (a una reparametrizzazione) alla loss di score matching denoising:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,v_0,\varepsilon}\left[\|u_\theta(v_t, t) - (v_0 - \varepsilon)\|_2^2\right] = c \cdot \mathcal{L}_{\text{DSM}}, \tag{11.9}$$

dove $c = c(t)$ è una costante dipendente dal tempo. L'errore di generazione del flow matching è quindi bounded dalla loss di score matching:

$$\mathrm{TV}(p_\theta^{\mathrm{FM}}, p_{\mathrm{data}}) \leq O(\mathcal{L}_{\mathrm{FM}}^{1/2}). \tag{11.10}$$

11.3 Analisi di Complessità Computazionale dei Modelli Trattati

Tabella di complessità: training e inferenza

Tabella 11.1: Complessità computazionale dei principali modelli del Volume IX.

Modello	Training	Inferenza	Memoria	Parallelizzabile
IQL (MARL)	$O(N \cdot P)$	$O(N \cdot P)$	$O(N \cdot P)$	Sì
QMIX	$O(N \cdot P + P_{\mathrm{mix}})$	$O(N \cdot \lvert A_i \rvert)$	$O(N \cdot P)$	Parziale
MAPPO	$O(N \cdot P + P_{\mathrm{crit}})$	$O(N \cdot P)$	$O(N \cdot P)$	Sì
MCTS-LLM	$O(P)$ pre-train	$O(N_{\mathrm{sim}} \cdot T \cdot P)$	$O(\mathrm{tree})$	Parziale
Best-of-N	$O(P)$ pre-train	$O(N \cdot T \cdot P)$	$O(N \cdot T)$	Sì
VDM	$O(K \cdot THW \cdot P)$	$O(K \cdot THW \cdot P)$	$O(THW \cdot d_z)$	Sì
VideoGPT	$O(N_{\mathrm{patch}}^2 \cdot P)$	$O(N_{\mathrm{patch}}^2 \cdot P)$	$O(K \cdot d_z)$	Parziale
CM-video	$O(K \cdot THW \cdot P)$	$O(1\text{–}4 \cdot THW \cdot P)$	$O(THW \cdot d_z)$	Sì
EWC	$O(P + \lvert D_1' \rvert \cdot P)$	$O(P)$	$O(P)$	Sì
GEM	$O(P + K \cdot M \cdot P)$	$O(P)$	$O(K \cdot M)$	No (QP)
A-GEM	$O(P + M \cdot P)$	$O(P)$	$O(M)$	Sì
PNN	$O(K \cdot P_{\mathrm{col}})$	$O(K \cdot P_{\mathrm{col}})$	$O(K \cdot P_{\mathrm{col}})$	Sì
VR-complex	$O(n^2 \cdot d)$ build	$O(n^3)$ omologia	$O(n^2)$	Parziale
Mapper	$O(n \cdot \lvert \mathcal{U} \rvert)$	—	$O(n)$	Sì

Legenda: P = parametri, N = agenti, K = passi diffusione, T = frame, H, W = risoluzione spaziale, M = buffer size, n = punti dati, d = dimensione, $\lvert D_1' \rvert$ = dataset Fisher subset.

Bottleneck computazionale in sistemi integrati

In un sistema agentivo completo che combina tutti i componenti del Volume IX:

- **Ragionamento** (Sezione B): $O(N_{\mathrm{sim}} \cdot T_{\mathrm{thought}} \cdot P_{\mathrm{LLM}})$ per MCTS con budget N_{sim} e pensiero di lunghezza T_{thought}.
- **Percezione video** (Sezione C): $O(s_T^{-1} s_S^{-2} \cdot THW \cdot P_{\mathrm{enc}})$ per l'encoder 3D-VAE con compressione.
- **Coordinazione multi-agente** (Sezione A): $O(N^2 \cdot P_{\mathrm{critic}})$ per il critico centralizzato MAPPO con N agenti.

- **Aggiornamento continual** (Sezione D): $O(M \cdot P_{\text{LLM}})$ per A-GEM con buffer M.

Il bottleneck è quasi sempre il LLM: $P_{\text{LLM}} \gg P_{\text{critic}}, P_{\text{enc}}$. Il costo totale di inferenza per un passo agentivo con video è quindi:

$$C_{\text{step}} = O\big(N_{\text{sim}} \cdot (T_{\text{thought}} + T_{\text{video}}/s_T s_S^2) \cdot P_{\text{LLM}}\big)\,. \tag{11.11}$$

11.4 Metodi Variazionali e Approssimazioni

Variational Inference per il POMDP

Il calcolo esatto della belief state b_t richiede $O(|S|)$ operazioni ad ogni passo. Per grandi spazi di stato, l'inference variazionale approssima b_t con una famiglia parametrica $q_\phi(\cdot) \in \mathcal{Q}$:

$$q_\phi^* = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q_\phi(\cdot) \| b_t(\cdot)). \tag{11.12}$$

Per la famiglia gaussiana $\mathcal{Q} = \{\mathcal{N}(\mu, \Sigma)\}$, la soluzione è la proiezione gaussiana della belief esatta:

$$\mu^* = \mathbb{E}_{b_t}[s], \qquad \Sigma^* = \mathbb{E}_{b_t}[(s-\mu^*)(s-\mu^*)^\top]. \tag{11.13}$$

Proposizione 11.5 (ELBO per il POMDP)**.** Il ELBO (Evidence Lower Bound) per l'aggiornamento della belief è:

$$\log P(o_{t+1} \mid o_{1:t}, a_{1:t}) \geq \mathbb{E}_{q_\phi(s)}[\log Z(o_{t+1}, s, a_t)] - \mathrm{KL}(q_\phi(\cdot) \| p(\cdot|o_{1:t}, a_{1:t})), \tag{11.14}$$

dove $Z(o_{t+1}, s, a_t) = \sum_{s'} Z(o_{t+1}|s', a_t) T(s'|s, a_t)$ è il likelihood dell'osservazione. Massimizzare l'ELBO rispetto a ϕ produce la migliore approssimazione variazionale della belief esatta.

Variational Autoencoder per la belief state

Gli LLM implementano implicitamente una forma di inference variazionale: il contesto c_t funge da sample $s \sim q_\phi(\cdot|c_t)$ dalla belief approssimata. La loss di pre-training (next-token prediction) è equivalente all'ELBO di un world model:

$$\mathcal{L}_{\text{LM}}(\theta) = -\mathbb{E}_{c_t}[\log \pi_\theta(o_{t+1}|c_t)] \geq -\log P(o_{t+1}|o_{1:t}, a_{1:t}). \tag{11.15}$$

Quindi l'LLM addestrato con next-token prediction apprende implicitamente un modello della dinamica del mondo (T) e della funzione di emissione (Z), anche senza supervisione esplicita sul POMDP.

Expectation-Maximization per il Continual Learning

L'algoritmo EM offre una prospettiva alternativa al continual learning. Modellando i task precedenti come variabili latenti $z_k \in \{1, \dots, K\}$:

$$P(\theta|\mathcal{D}_{1:T}) \propto \prod_{k=1}^{T} P(\mathcal{D}_k|\theta) \cdot P(\theta). \tag{11.16}$$

L'E-step stima il peso di ogni task, l'M-step ottimizza θ con i pesi fissi:

$$\text{(E-step)} \quad r_k(\theta^{(t)}) = P(\mathcal{D}_k|\theta^{(t)}) / \sum_{k'} P(\mathcal{D}_{k'}|\theta^{(t)}), \tag{11.17}$$

$$\text{(M-step)} \quad \theta^{(t+1)} = \arg\max_{\theta} \sum_{k} r_k(\theta^{(t)}) \log P(\mathcal{D}_k|\theta) + \log P(\theta). \tag{11.18}$$

Proposizione 11.6 (EM per CL converge al MAP sequenziale)**.** L'EM applicato al continual learning con $P(\theta) = \mathcal{N}(\theta_1^*, F_1^{-1})$ converge alla stima MAP del posteriore congiunto $P(\theta|\mathcal{D}_{1:T})$, che è esattamente il punto fisso di EWC esteso a T task.

11.5 Metriche di Valutazione per il Continual Learning

Forward Transfer e Backward Transfer

Definizione 11.3 (Forward Transfer)**.** Il *forward transfer* al passo t misura quanto i task precedenti hanno facilitato l'apprendimento del task corrente:

$$\text{FWT}_t = \frac{1}{t-1} \sum_{k=1}^{t-1} \left[\mathcal{L}_k(f_\theta^{(0)}) - \mathcal{L}_k(f_\theta^{(t)}) \right], \tag{11.19}$$

dove $f_\theta^{(0)}$ è il modello iniziale (non addestrato) e $f_\theta^{(t)}$ è quello dopo t task. Un $\text{FWT}_t > 0$ indica transfer positivo dai task precedenti al task t; negativo indica *negative transfer* (i task precedenti ostacolano l'apprendimento).

Definizione 11.4 (Backward Transfer)**.** Il *backward transfer* al passo t misura l'impatto dei task nuovi sulle performance dei task vecchi:

$$\text{BWT}_t = \frac{1}{t-1} \sum_{k=1}^{t-1} \left[\mathcal{L}_k(f_\theta^{(t)}) - \mathcal{L}_k(f_\theta^{(k)}) \right]. \tag{11.20}$$

Un $\text{BWT}_t < 0$ indica miglioramento retroattivo (backward transfer positivo); un

$\mathrm{BWT}_t > 0$ indica catastrophic forgetting. Il forgetting score FGT è $\mathrm{FGT}_t = \max(0, \mathrm{BWT}_t)$.

Curva di apprendimento e intransigenza

Definizione 11.5 (Intransigenza)**.** L'*intransigenza* è la difficoltà del modello ad apprendere nuovi task a causa di un bias acquisito dai task precedenti:

$$I_t = \mathcal{L}_t(f_\theta^{(t)}) - \mathcal{L}_t(f_{\theta_t^*}), \tag{11.21}$$

dove $f_{\theta_t^*}$ è il modello addestrato solo su task t dall'inizio. Un'alta intransigenza indica che il modello "fatica" ad adattarsi al task t a causa dei vincoli imposti dai task precedenti.

Proposizione 11.7 (Trade-off forgetting-intransigenza)**.** Per un modello con capacità fissa P, esiste un trade-off fondamentale:

$$\mathrm{FGT}_t + I_t \geq \frac{C(P)}{t}, \tag{11.22}$$

dove $C(P) > 0$ dipende dalla capacità del modello. Un modello non può simultaneamente avere basso forgetting e bassa intransigenza su t task con capacità finita.

Dimostrazione. [Sketch] Il bound segue dalla disuguaglianza di capacità di Shannon: il modello deve codificare le informazioni di tutti i t task in P parametri. Con B bit di informazione per task, il totale è tB, ma il modello ne può contenere solo $C(P) = O(P \log P)$. Il forgetting è proporzionale all'informazione persa dai task vecchi, l'intransigenza all'informazione non acquisita dal task nuovo. La loro somma è bounded inferiormente dalla frazione di informazione non rappresentabile. □

11.6 Reti Neurali su Grafi e Topologia

Graph Neural Networks e omologia dei grafi

Le *Graph Neural Networks* (GNN) operano su grafi $G = (V, E)$, aggregando informazioni dai vicini. La connessione con la topologia è diretta: il grafo è un complesso simpliciale di dimensione 1, e le sue proprietà topologiche (componenti connesse, cicli) influenzano il comportamento delle GNN.

Definizione 11.6 (Message Passing)**.** Lo strato di message passing di una GNN aggiorna le rappresentazioni dei nodi:

$$h_v^{(l+1)} = \mathrm{UPDATE}\big(h_v^{(l)}, \mathrm{AGGREGATE}\big(\{h_u^{(l)} : u \in \mathcal{N}(v)\}\big)\big), \tag{11.23}$$

dove $\mathcal{N}(v)$ è il vicinato del nodo v.

Teorema 11.3 (Espressività delle GNN e test di isomorfismo di Weisfeiler-Lehman)**.** Una GNN con message passing L layer può distinguere due grafi non isomorfi solo se il *Weisfeiler-Lehman graph isomorphism test* di ordine L li distingue. Quindi la GNN standard non riesce a distinguere grafi con la stessa struttura WL (es. grafi regolari con lo stesso numero di nodi e archi).

Topological GNN: incorporare l'omologia

Per superare il limite del test WL, le *Topological GNN* incorporano informazioni di omologia di ordine superiore:

$$h_v^{(l+1)} = \text{UPDATE}\Big(h_v^{(l)},\ \text{AGG}_{\text{edge}}\big(\{h_e^{(l)}\}_{e\ni v}\big)\,,\ \text{AGG}_{\text{tri}}\Big(\{h_t^{(l)}\}_{t\ni v}\Big)\Big)\,, \tag{11.24}$$

dove $h_e^{(l)}$ sono le rappresentazioni degli archi (1-simplessi) e $h_t^{(l)}$ quelle dei triangoli (2-simplessi) che contengono v.

Proposizione 11.8 (Topological GNN distingue più grafi)**.** La Topological GNN che usa simplessici fino a dimensione k può distinguere tutti i grafi distinguibili dal test WL di ordine k, più alcuni grafi indistinguibili al test WL standard (ordine 1).

La connessione con la TDA del Capitolo 5 è diretta: i numeri di Betti $\beta_0, \beta_1, \beta_2$ del grafo (componenti connesse, cicli, cavità) sono calcolabili tramite omologia simpliciale e possono essere usati come feature globali aggiuntive per le GNN.

Persistent Homology come pooling globale

Definizione 11.7 (Topological Pooling)**.** Per un grafo G con rappresentazioni dei nodi $\{h_v\}$, il *topological pooling* costruisce il complesso VR sulle rappresentazioni e calcola il diagramma di persistenza:

$$\text{TopPool}(G, \{h_v\}) = \text{Dgm}_k(\{h_v : v \in V\}). \tag{11.25}$$

Il diagramma viene poi convertito in un vettore (persistence image o landscape) e usato come rappresentazione globale del grafo per la classificazione.

Questo approccio è invariante rispetto alla permutazione dei nodi (per costruzione della TDA) e cattura strutture topologiche multi-scala che il message passing non riesce a rappresentare.

Capitolo 12

Implementazione e Considerazioni Pratiche

12.1 Implementazione di GRPO in Pseudo-Codice

Algorithm 10 GRPO: Group Relative Policy Optimization

Require: Modello $\pi_{\theta^{\text{old}}}$, riferimento π_{ref}, dataset $\mathcal{D}$, verifier R, iperparametri $\varepsilon, \beta, G, \eta$

1: **for** ogni batch di problemi $\{x_1, \ldots, x_B\} \sim \mathcal{D}$ **do**
2: **for** $i = 1, \ldots, B$ **do**
3: Campiona G completamenti: $(T_j^i, y_j^i) \sim \pi_{\theta^{\text{old}}}(\cdot|x_i)$, $j = 1, \ldots, G$
4: Valuta ricompense: $R_j^i = R(x_i, y_j^i) \in \{0, 1\}$
5: Calcola vantaggi normalizzati: $\hat{A}_j^i = (R_j^i - \bar{R}^i)/\max(\text{std}(R^i), 10^{-8})$
6: **end for**
7: Calcola la loss GRPO sul batch:

$$\mathcal{L} = -\frac{1}{B \cdot G} \sum_{i=1}^{B} \sum_{j=1}^{G} \min\Big(r_j^i \hat{A}_j^i,\ \text{clip}(r_j^i, 1-\varepsilon, 1+\varepsilon) \hat{A}_j^i \Big) + \beta \cdot \frac{1}{B} \sum_{i=1}^{B} \text{KL}(\pi_\theta(\cdot|x_i) \| \pi_{\text{ref}}(\cdot|x_i))$$

dove $r_j^i = \pi_\theta(T_j^i, y_j^i|x_i)/\pi_{\theta^{\text{old}}}(T_j^i, y_j^i|x_i)$

8: Aggiorna: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
9: **end for**

Calcolo efficiente del ratio di importanza

Il ratio $r_j^i = \pi_\theta(T_j^i, y_j^i|x_i)/\pi_{\theta^{\text{old}}}(T_j^i, y_j^i|x_i)$ si calcola in log-spazio per stabilità numerica:

$$\log r_j^i = \sum_{t=1}^{|T_j^i|+|y_j^i|} \left[\log \pi_\theta(w_t|w_{<t}, x_i) - \log \pi_{\theta^{\text{old}}}(w_t|w_{<t}, x_i) \right], \tag{12.1}$$

dove w_t sono i token del completamento (T_j^i, y_j^i). Il log-ratio viene poi esponenziato: $r_j^i = e^{\log r_j^i}$. Il clipping viene applicato al ratio esponenziato.

Stabilità numerica e trucchi implementativi

1. **Normalizzazione del vantaggio**: il $\max(\text{std}(R), 10^{-8})$ nel denominatore evita la divisione per zero quando tutte le ricompense del gruppo sono uguali (es. tutti 0 o tutti 1).

2. **KL approssimata**: la KL $\mathrm{KL}(\pi_\theta \| \pi_{\text{ref}})$ è approssimata con il *k3 estimator* (Schulman, 2020): $\hat{\mathrm{KL}} \approx (r-1) - \log r$ (con $r = \pi_\theta / \pi_{\text{ref}}$), che è sempre non-negativo e più stabile dell'estimatore diretto.

3. **Gradient clipping**: i gradienti vengono clippati a norma ≤ 1.0 prima dell'aggiornamento per evitare esplosioni.

4. **Reference model freeze**: π_{ref} non viene aggiornato durante il training GRPO; vengono salvati i log-probability di π_{ref} sui completamenti per evitare un forward pass aggiuntivo.

12.2 Implementazione di EWC per Transformer

Calcolo della Fisher su un Transformer

Algorithm 11 Fisher Diagonale per Transformer

Require: Modello f_θ (Transformer), dataset $\mathcal{D}'_1$, θ^*_1

1: $F \leftarrow \mathbf{0} \in \mathbb{R}^P$ ▷ Fisher diagonale inizializzata a zero
2: **for** ogni minibatch $(x, y) \in \mathcal{D}'_1$ **do**
3: Forward pass: $\hat{y} = f_{\theta^*_1}(x)$
4: Campiona $\tilde{y} \sim P(\cdot|x, \theta^*_1)$ ▷ campionamento invece di argmax
5: Calcola loss: $\ell = -\log P(\tilde{y}|x, \theta^*_1)$
6: Backward pass: $g = \nabla_\theta \ell$
7: Accumula: $F \leftarrow F + g^2/|\mathcal{D}'_1|$
8: **end forreturn** F ▷ Fisher diagonale normalizzata

Nota sul campionamento vs argmax

L'uso del *campionamento* (riga 4) invece dell'argmax nella computazione della Fisher è essenziale: la Fisher è l'aspettativa sotto la distribuzione del modello, non sotto la distribuzione dei dati. L'argmax produce una stima distorta della Fisher (*Generalized Gauss-Newton matrix*, GGN), mentre il campionamento produce la Fisher esatta. Per la loss di cross-entropy con etichette one-hot, le due coincidono solo se il modello è perfetto ($\hat{y} = y$).

Integrazione di EWC con LoRA

LoRA (Hu et al., 2022) fine-tunes solo le perturbazioni $\Delta W_k = B_k A_k$ (rango ridotto) lasciando i pesi pre-addestrati W^0_k fissi. La loss EWC con LoRA si modifica:

$$\mathcal{L}_{\text{EWC-LoRA}} = \mathcal{L}_2(W^0 + \Delta W) + \frac{\lambda}{2} \sum_k \left\| F^{1/2}_{1,k} \odot \Delta W_k \right\|^2_F , \tag{12.2}$$

dove $F^{1/2}_{1,k}$ è la radice quadrata della Fisher diagonale del layer k. Questo vincola le perturbazioni LoRA a rimanere piccole nelle direzioni importanti per il task 1, garantendo contemporaneamente che i pesi pre-addestrati non vengano modificati (protezione architettonica dal forgetting).

12.3 Considerazioni sull'Hardware e la Scalabilità

Multi-GPU training per VDM

Il Video Diffusion Model richiede tensori $v_k \in \mathbb{R}^{B\times T\times H\times W\times C}$ (con B = batch size). Per $B = 4$, $T = 16$, $H = W = 256$, $C = 4$ (latent channels), la memoria è $4 \times 16 \times 256 \times 256 \times 4 \times 4$ byte $= 4$ GB solo per il batch. Con i gradienti e l'optimizer state (Adam), il costo totale è ≈ 20 GB.

Le strategie di distribuzione su multi-GPU includono:

- **Data Parallelism**: ogni GPU processa un sottoinsieme del batch. Overhead: all-reduce dei gradienti $O(P/B_{\text{GPU}})$.
- **Tensor Parallelism**: ogni GPU contiene una partizione dei pesi. Overhead: all-reduce delle attivazioni $O(B \cdot THW \cdot d/N_{\text{GPU}})$.
- **Pipeline Parallelism**: i layer sono distribuiti su GPU diverse. Overhead: bolle di pipeline $O(N_{\text{GPU}}/B)$.

Per VDM, il tensor parallelism sull'asse d_z (canali latenti) è particolarmente efficiente perché le convoluzioni 3D si fattorizzano naturalmente.

Inference ottimizzata per MCTS con LLM

Il collo di bottiglia di MCTS con LLM è il KV-cache: ad ogni nodo dell'albero, il prefisso del ragionamento ha un KV-cache diverso. Con N_{sim} simulazioni e profondità D, il costo totale di memoria è $O(N_{\text{sim}} \cdot D \cdot L \cdot d_{\text{head}})$ dove L è il numero di layer e d_{head} la dimensione delle head.

Tecniche di ottimizzazione:

1. **KV-cache sharing**: i nodi che condividono un prefisso comune condividono il KV-cache fino al punto di divergenza. Risparmio: O(profondità comune).
2. **Speculative decoding**: si genera un draft rapido (modello piccolo) e si verifica con il modello grande, riducendo il numero di forward pass.
3. **Batched MCTS**: si raggruppano le espansioni di nodi diversi in un unico batch forward pass, sfruttando il parallelismo del GPU.

Capitolo 13

Esercizi e Problemi Aperti

13.1 Esercizi con Soluzione

Esercizio 1: Aggiornamento della belief state

Problema. Considera un POMDP con $S = \{s_1, s_2\}$, $A = \{a\}$ (azione unica), $O = \{o_1, o_2\}$ e i seguenti parametri:

$$\begin{aligned} T(s_1|s_1, a) &= 0.9, \quad T(s_2|s_1, a) = 0.1, \\ T(s_1|s_2, a) &= 0.2, \quad T(s_2|s_2, a) = 0.8, \\ Z(o_1|s_1, a) &= 0.8, \quad Z(o_2|s_1, a) = 0.2, \\ Z(o_1|s_2, a) &= 0.3, \quad Z(o_2|s_2, a) = 0.7. \end{aligned}$$

Partendo dalla belief $b_0 = (0.5, 0.5)$, calcola b_1 dopo l'osservazione o_1.

Soluzione. Prima calcoliamo il predittore (belief dopo la transizione, prima dell'osservazione):

$$\begin{aligned} \hat{b}_1(s_1) &= T(s_1|s_1, a)\, b_0(s_1) + T(s_1|s_2, a)\, b_0(s_2) = 0.9 \cdot 0.5 + 0.2 \cdot 0.5 = 0.55, \\ \hat{b}_1(s_2) &= T(s_2|s_1, a)\, b_0(s_1) + T(s_2|s_2, a)\, b_0(s_2) = 0.1 \cdot 0.5 + 0.8 \cdot 0.5 = 0.45. \end{aligned}$$

Poi aggiorniamo con l'osservazione o_1:

$$\begin{aligned} b_1^{\text{un}}(s_1) &= Z(o_1|s_1, a)\, \hat{b}_1(s_1) = 0.8 \cdot 0.55 = 0.44, \\ b_1^{\text{un}}(s_2) &= Z(o_1|s_2, a)\, \hat{b}_1(s_2) = 0.3 \cdot 0.45 = 0.135, \\ Z &= 0.44 + 0.135 = 0.575. \end{aligned}$$

La belief aggiornata è: $b_1 = (0.44/0.575,\ 0.135/0.575) \approx (0.765,\ 0.235)$. L'osservazione o_1 (più probabile in s_1) ha spostato la belief verso s_1. □

Esercizio 2: Shapley value in un gioco a tre agenti

Problema. Considera un gioco di coalizione con $\mathcal{N} = \{1, 2, 3\}$ e funzione di valore $v(\emptyset) = 0$, $v(\{1\}) = 3$, $v(\{2\}) = 2$, $v(\{3\}) = 1$, $v(\{1, 2\}) = 6$, $v(\{1, 3\}) = 5$, $v(\{2, 3\}) = 4$, $v(\{1, 2, 3\}) = 9$. Calcola lo Shapley value di ciascun agente.

Soluzione. Ci sono $3! = 6$ ordinamenti possibili degli agenti. Per ciascuno, il contributo marginale di ogni agente è $v(S \cup \{i\}) - v(S)$:

Ordinamento	ϕ_1	ϕ_2	ϕ_3
$(1,2,3)$	$v(\{1\}) - 0 = 3$	$v(\{1,2\}) - v(\{1\}) = 3$	$v(\mathcal{N}) - v(\{1,2\}) =$
$(1,3,2)$	3	$v(\mathcal{N}) - v(\{1,3\}) = 4$	$v(\{1,3\}) - v(\{1\}) =$
$(2,1,3)$	$v(\{1,2\}) - v(\{2\}) = 4$	2	$v(\mathcal{N}) - v(\{1,2\}) =$
$(2,3,1)$	$v(\mathcal{N}) - v(\{2,3\}) = 5$	2	$v(\{2,3\}) - v(\{2\}) =$
$(3,1,2)$	$v(\{1,3\}) - v(\{3\}) = 4$	$v(\mathcal{N}) - v(\{1,3\}) = 4$	1
$(3,2,1)$	$v(\mathcal{N}) - v(\{2,3\}) = 5$	$v(\{2,3\}) - v(\{3\}) = 3$	1
Media	$(3+3+4+5+4+5)/6 = 4$	$(3+4+2+2+4+3)/6 = 3$	$(3+2+3+2+1+1)$

Quindi $\phi = (4, 3, 2)$. Verifica di efficienza: $4 + 3 + 2 = 9 = v(\mathcal{N})$. □

Esercizio 3: Omologia di un toro

Problema. Calcola i numeri di Betti del toro $T^2 = S^1 \times S^1$.

Soluzione. Il toro si triangola con 9 vertici, 27 spigoli e 18 triangoli. Usando il teorema di Künneth per l'omologia:

$$H_k(T^2; \mathbb{F}) \cong \bigoplus_{i+j=k} H_i(S^1; \mathbb{F}) \otimes H_j(S^1; \mathbb{F}). \tag{13.1}$$

Poiché $H_0(S^1) = \mathbb{F}$, $H_1(S^1) = \mathbb{F}$, $H_k(S^1) = 0$ per $k \geq 2$:

$$\begin{aligned} H_0(T^2) &= H_0(S^1) \otimes H_0(S^1) = \mathbb{F}, & \beta_0 &= 1, \\ H_1(T^2) &= H_0(S^1) \otimes H_1(S^1) \oplus H_1(S^1) \otimes H_0(S^1) = \mathbb{F}^2, & \beta_1 &= 2, \\ H_2(T^2) &= H_1(S^1) \otimes H_1(S^1) = \mathbb{F}, & \beta_2 &= 1. \end{aligned}$$

Caratteristica di Eulero: $\chi = 1 - 2 + 1 = 0$. Questo riflette il fatto che il toro è una superficie chiusa orientabile di genere 1. □

Esercizio 4: Convergenza di Best-of-N

Problema. Un LLM risolve problemi matematici con probabilità $p = 0.05$ per singolo tentativo. Quanti campioni N servono affinché la probabilità di successo superi 0.95? Confronta con l'aumento di modello che riduce la loss da $\mathcal{L}_0$ a $\mathcal{L}_0 - \Delta\mathcal{L}$.

Soluzione. Per best-of-N: $P(\text{successo}) = 1 - (1 - 0.05)^N \geq 0.95$.

$$(0.95)^N \leq 0.05 \implies N \geq \frac{\ln(0.05)}{\ln(0.95)} = \frac{-2.996}{-0.051} \approx 58.4,$$

quindi $N^* = 59$ campioni. Il compute totale è $59 \times C_{\text{single}}$.

Per confronto, con le scaling laws di Chinchilla, aumentare il compute di addestramento di un fattore k riduce la loss di $\Delta\mathcal{L} \propto k^{-0.5}$. La probabilità di successo per singolo tentativo cresce approssimativamente come $p(C_{\text{train}}) \approx p_0 \cdot (C_{\text{train}}/C_0)^{\alpha}$, con $\alpha \approx 0.1$–0.3. Per arrivare a $p = 0.95$ in un singolo tentativo:

$$k = (0.95/0.05)^{1/\alpha} = 19^{1/0.2} \approx 6.7 \times 10^5$$

ovvero si deve aumentare il compute di addestramento di 670,000 volte. Il best-of-59 richiede solo 59 volte più compute di inferenza, che equivale (per un modello con n_q query) a $59/n_q$ del compute di training. Per $n_q = 1000$ query, il best-of-N è $\approx 11,000$ volte più efficiente dell'aumento di modello. □

13.2 Problemi Aperti

Problema aperto 1: Scaling del test-time compute con MCTS

Le leggi di scaling del training compute (Chinchilla) sono ben caratterizzate empiricamente. Per il test-time compute, la situazione è meno chiara: l'esponente α_{test} nella legge $\text{Err} \propto C_{\text{test}}^{-\alpha_{\text{test}}}$ dipende fortemente dal tipo di problema e dalla strategia (best-of-N, MCTS, beam search).

Domanda aperta: esiste una legge di scaling universale per il test-time compute che dipende solo dalla "difficoltà" del problema (definita in modo information-theoretic) e non dalla strategia specifica?

Progresso recente: alcuni risultati (Snell et al., 2024) suggeriscono che la strategia ottimale di test-time compute dipende dal regime:

- Per p moderato ($p > 0.1$): best-of-N è quasi ottimale.
- Per p piccolo ($p < 0.01$): MCTS con PRM domina best-of-N.
- Per p molto piccolo ($p < 10^{-3}$): beam search (esplorazione sistematica) supera MCTS (esplorazione stocastica).

Un'unificazione teorica di questi regimi rimane aperta.

Problema aperto 2: Forgetting catastrofico nei Transformer

La teoria del forgetting (Sezione D, §530) si basa sull'approssimazione quadratica (4.3) della loss. Per reti profonde, questa approssimazione è accurata solo in un intorno piccolo di θ_1^*, mentre il fine-tuning su task 2 può allontanarsi significativamente.

Domanda aperta: esiste un bound non-quadratico sul forgetting per reti neurali profonde che sia sia teoricamente fondato sia praticabile?

Connessione con la TDA: la topologia del loss landscape (Sezione E, §539) suggerisce che il forgetting dipende dalla connettività dei minimi: se θ_1^* e θ_2^* sono nello stesso bacino di attrazione, il forgetting è limitato. Formalizzare questa intuizione tramite persistent homology è un problema aperto.

Problema aperto 3: Mean-Field Games per LLM multi-agente

I mean-field games (Sezione A, §518) forniscono il limite termodinamico del MARL, ma la loro applicazione a sistemi di LLM è non banale: gli agenti LLM hanno osservazioni e azioni in spazi di testo ad alta dimensione, violando le ipotesi di regolarità standard dei MFG.

Domanda aperta: come si estende la teoria dei mean-field games a spazi di azione discreti e ad alta dimensione (testo) con funzioni di transizione non Markoviane (memoria contestuale)?

Approcci parziali: MFG con spazi discreti (Guo et al., 2019) richiede la differenziabilità dell'Hamiltoniano rispetto alla misura μ_t, che nel caso del testo equivale a differenziabilità rispetto alle distribuzioni sui vocabolari. Questo è collegato alla differenziabilità del softmax e alle tecniche di *straight-through estimator*.

Problema aperto 4: Interpretazione topologica della generalizzazione

La Proposizione sul bound di generalizzazione topologico (Capitolo 6, Sezione 4) è un bound PAC-Bayes approssimato: non è stato derivato rigorosamente da principi primi.

Domanda aperta: esiste un bound di generalizzazione rigoroso in termini di invarianti topologici (numeri di Betti, persistenza totale) delle rappresentazioni neurali, derivabile dai principi del PAC-learning?

Connessione: il principio dell'information bottleneck (Vol. IV, §213) fornisce un bound di generalizzazione in termini dell'informazione mutua $I(Z;X)$ tra le rappresentazioni Z e l'input X. La connessione con la persistenza topologica (suggerita nella §542) potrebbe portare a un bridge formale tra TDA e teoria PAC.

Problema aperto 5: Convergenza globale di MAPPO in giochi non cooperativi

La convergenza di MAPPO (Sezione A, §516) è dimostrata solo per il caso cooperativo ($R_i = R$ per ogni i). Per giochi a somma generale, la convergenza è aperta.

Domanda aperta: per quali classi di giochi a somma generale con più agenti MAPPO converge a un equilibrio di Nash? Esistono condizioni sufficienti in termini della struttura del gioco (es. monotonia del migliore risposta, giochi a potenziale) che garantiscono la convergenza?

Evidenza parziale: per giochi a due giocatori a somma zero, MAPPO empiricamente oscilla invece di convergere (cicli di best-response). Per giochi a potenziale (Monderer e Shapley, 1996), la convergenza al Nash puro è garantita dalla dinamica best-response (§518), ma non è chiaro se MAPPO approssimi questa dinamica.

Connessione con la TDA: la topologia delle traiettorie di addestramento di MAPPO (nello spazio dei parametri) potrebbe rivelare se l'algoritmo si comporta come un sistema conservativo (traiettorie chiuse, indicando oscillazione) o dissipativo (traiettorie che convergono). La persistent homology di queste traiettorie è un tool potenziale per distinguere i due regimi.

Problema aperto 6: SMC ottimale per LLM con PRM imperfetto

Il Sequential Monte Carlo per LLM (§algoritmi avanzati) usa il PRM come funzione di peso V_ϕ. Quando V_ϕ è imperfetto (il caso reale), la distribuzione target $\pi^* \propto \pi_\theta \cdot V_\phi$ differisce dalla distribuzione vera $\pi^*_{\text{true}} \propto \pi_\theta \cdot R$ (con ricompensa binaria vera).

Domanda aperta: qual è la politica di campionamento SMC ottimale (scelta della proposta q, frequenza di resampling, numero di particelle) che minimizza l'errore di approssimazione dovuto all'imprecisione del PRM?

Formalizzazione: sia $\varepsilon_{\text{PRM}} = \|V_\phi - R\|_\infty$ l'errore del PRM. L'errore di stima di SMC è:

$$\|\hat{\pi}_{\text{SMC}} - \pi^*_{\text{true}}\|_{\text{TV}} \leq O\left(\varepsilon_{\text{PRM}} + \frac{C_T}{N}\right),$$

dove C_T dipende dalla proposta. La proposta ottimale minimizza C_T per un dato ε_{PRM} e N. Trovare questa proposta esplicitamente rimane aperto.

Indice delle Sezioni

§	Titolo	Sezione
511	Processi di Decisione Markoviani Parzialmente Osservabili	A
512	ReAct: Ragionamento e Azione Interlacciati	A
513	Tool-Use Formale: Funzioni Esterne come Estensione dell'Azione	A
514	Giochi a Somma Generale: Definizioni e Notazione	A
515	Equilibrio di Nash: Esistenza e Caratterizzazione	A
516	Algoritmi MARL: IQL, QMIX e MAPPO	A
517	Teoria dei Giochi Cooperativa: Shapley Value e Credito	A
518	Equilibri di Nash, Potenziale e Mean-Field Games	A
519	La Dicotomia Training-Time vs Test-Time Compute	B
520	Monte Carlo Tree Search per LLM	B
521	Process Reward Models	B
522	Best-of-N con Verifier: Analisi Statistica	B
523	Beam Search come Caso Speciale di MCTS	B
524	Thinking Models: RL sul Ragionamento e GRPO	B
525	Rappresentazione Spaziotemporale del Video	C
526	Video Diffusion Models: Derivazione Completa	C
527	Sora e Diffusion Transformer Spaziotemporale	C
528	VideoGPT: Generazione Autoregressiva su Token Video	C
529	Consistency Models e Flow Matching per Video	C
530	Il Problema del Forgetting: Definizione Formale	D
531	Teoria PAC su Distribuzioni Non Stazionarie	D
532	Elastic Weight Consolidation: Derivazione Bayesiana	D
533	Progressive Neural Networks e PackNet	D
534	GEM e A-GEM: Gradient Episodic Memory	D
535	Complessi Simpliciali: Definizioni Fondamentali	E
536	Omologia: Cicli, Bordi e Spazi di Betti	E
537	Omologia Persistente: il Teorema di Struttura	E
538	Stabilità del Diagramma di Persistenza	E
539	Numeri di Betti dei Loss Landscape	E
540	Il Mapper Algorithm: Visualizzazione Topologica	E
541	Distanze tra Rappresentazioni Neurali tramite TDA	E
542	Persistent Homology per lo Studio del Grokking	E

Glossario dei Simboli Principali

Simbolo	**Significato**	**Introdotto in**
Sezione A — AI Agentiva		
$M = (S, A, O, T, Z, R, \gamma)$	POMDP	§511
$b_t \in \Delta(S)$	belief state	§511
$\tau = (o_1, r_1, a_1, \ldots)$	traiettoria ReAct	§512
π_θ	politica dell'agente LLM	§512
$f_k : X_k \to Y_k$	strumento (tool)	§513
$\mathcal{G} = (\mathcal{N}, S, \{A_i\}, T, \{R_i\}, \gamma)$	gioco a N agenti	§514
$\pi^* = (\pi_1^*, \ldots, \pi_N^*)$	equilibrio di Nash	§515
$Q_{\text{tot}}(s, \mathbf{a})$	valore congiunto QMIX	§516
$\phi_i(v)$	Shapley value dell'agente i	§517
$\Phi : S \times A \to \mathbb{R}$	potenziale del gioco	§518
(μ_t, V)	soluzione del mean-field game	§518
Sezione B — Inference-Time Compute		
C_{test}	test-time compute budget	§519
α_{test}	esponente di scaling test-time	§519
$\text{UCB1}(p, a)$	punteggio UCB1 per MCTS	§520
$V_\phi^{\text{PRM}}(x, r_{\le t})$	Process Reward Model	§521
$y^* = \arg\max_i V_\phi(x, y_i)$	Best-of-N con verifier	§522
$\hat{A}_i$	vantaggio normalizzato GRPO	§524
$r_i = \pi_\theta / \pi_{\theta^{\text{old}}}$	ratio di importanza	§524
Sezione C — Generazione Video		
$v : \{1, \ldots, T\} \times \{1, \ldots, H\} \times \{1, \ldots, W\} \to \mathbb{R}^3$	video	§525
(E_ϕ, D_ϕ)	video tokenizer (3D-VAE)	§525
$\bar{\alpha}_k = \prod_{j=1}^{k}(1 - \beta_j)$	schedula di diffusione	§526
$\epsilon_\theta(v_k, k, c)$	denoiser DDPM per video	§526
$\text{PE}(t, h, w)$	positional encoding 3D	§527
$k^* = \arg\min_k \lVert z_i - e_k \rVert_2$	quantizzatore VQ	§528
$f_\theta(v_k, k)$	consistency function video	§529
$u_\theta(v_t, t)$	campo di velocità (flow matching)	§529
Sezione D — Continual Learning		
FGT_t	forgetting score al passo t	§530
$F_1 = \nabla_\theta^2 \mathcal{L}_1 \vert_{\theta_1^*}$	matrice di Fisher del task 1	§532
$\mathcal{L}_{\text{EWC}}(\theta)$	loss EWC	§532
$h^{(k,l)}$	attivazione colonna k, strato l	§533
$U^{(k,j,l)}$	lateral connections PNN	§533
$\tilde{g}$	gradiente proiettato GEM	§534
g_{ref}	gradiente medio A-GEM	§534
Sezione E — Topologia Applicata		
$\sigma = [v_0, \ldots, v_k]$	k-simplesso	§535
$\partial_k : C_k \to C_{k-1}$	operatore bordo	§536
$H_k(K; \mathbb{F}) = \ker \partial_k / \operatorname{im} \partial_{k+1}$	k-esimo gruppo di omologia	§536
$\beta_k = \dim H_k$	k-esimo numero di Betti	§536
$\text{VR}(\mathcal{X}, \varepsilon)$	complesso di Vietoris-Rips	§537
$\text{Dgm}_k(\mathcal{X})$	diagramma di persistenza	§537
$d_\infty(A, B)$	distanza bottleneck	§538
G_{Mapper}	grafo output del Mapper	§540
Simboli generali		
$\Delta(S)$	simplesso delle distribuzioni su S	Vol. I
$\text{KL}(P \Vert Q)$	divergenza KL	Vol. II

Bibliografia

Aumann, R. J. (1987). Correlated equilibrium as an expression of Bayesian rationality. *Econometrica*, 55(1):1–18.

Hart, S. e Mas-Colell, A. (2000). A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150.

Huang, M., Malhamé, R. P. e Caines, P. E. (2006). Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Communications in Information and Systems*, 6(3):221–252.

Kakade, S. (2001). A natural policy gradient. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 14.

Kaelbling, L. P., Littman, M. L. e Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134.

Lasry, J.-M. e Lions, P.-L. (2007). Mean field games. *Japanese Journal of Mathematics*, 2(1):229–260.

Monderer, D. e Shapley, L. S. (1996). Potential games. *Games and Economic Behavior*, 14(1):124–143.

Nash, J. (1951). Non-cooperative games. *Annals of Mathematics*, 54(2):286–295.

Rashid, T., Samvelyan, M., Schroeder de Witt, C., Farquhar, G., Foerster, J. e Whiteson, S. (2018). QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Shapley, L. S. (1953). A value for n-person games. In Kuhn, H. W. e Tucker, A. W. (eds.), *Contributions to the Theory of Games*, volume 2, pp. 307–317. Princeton University Press.

Wang, J., Ren, Z., Liu, T., Yu, Y. e Zhang, C. (2020). QPLEX: Duplex dueling multi-agent Q-learning. In *International Conference on Learning Representations (ICLR)*.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. e Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A. e Wu, Y. (2022). The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35.

Coulom, R. (2006). Efficient selectivity and backup operators in Monte-Carlo tree search. In *International Conference on Computers and Games*.

Kocsis, L. e Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *European Conference on Machine Learning (ECML)*.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I. e Cobbe, K. (2023). Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

OpenAI (2024). OpenAI o1 technical report. Technical report, OpenAI.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y. e Guo, D. (2024). DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Snell, C., Lee, J., Xu, K. e Kumar, A. (2024). Scaling LLM test-time compute optimally is more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Song, Y., Dhariwal, P., Chen, M. e Sutskever, I. (2023). Consistency models. In *International Conference on Machine Learning (ICML)*.

Ho, J., Jain, A. e Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M. e Fleet, D. J. (2022). Video diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M. e Le, M. (2022). Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Liu, X., Gong, C. e Liu, Q. (2022). Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

OpenAI (2024b). Sora: Video generation models as world simulators. Technical report, OpenAI. https://openai.com/sora.

Peebles, W. e Xie, S. (2023). Scalable diffusion models with transformers. In *International Conference on Computer Vision (ICCV)*.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S. e Poole, B. (2021). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*.

Yan, W., Zhang, Y., Abbeel, P. e Srinivas, A. (2021). VideoGPT: Video generation using VQ-VAE and transformers. *arXiv preprint arXiv:2104.10157*.

Chaudhry, A., Ranzato, M., Rohrbach, M. e Elhoseiny, M. (2019). Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations (ICLR)*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D. e Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13):3521–3526.

Lopez-Paz, D. e Ranzato, M. (2017). Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30.

Mallya, A. e Lazebnik, S. (2018). PackNet: Adding multiple tasks to a single network by iterative pruning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G. e Lampert, C. H. (2017). iCaRL: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R. e Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R. e Hadsell, R. (2018). Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning (ICML)*.

Zenke, F., Poole, B. e Ganguli, S. (2017). Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*.

Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F. e Ziegelmeier, L. (2017). Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research (JMLR)*, 18(8):1–35.

Bubenik, P. (2015). Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research (JMLR)*, 16(1):77–102.

Cohen-Steiner, D., Edelsbrunner, H. e Harer, J. (2007). Stability of persistence diagrams. *Discrete and Computational Geometry*, 37(1):103–120.

Edelsbrunner, H. e Harer, J. (2010). *Computational Topology: An Introduction*. American Mathematical Society.

Gotmare, A., Keskar, N. S., Xiong, C. e Socher, R. (2018). Using mode connectivity for loss landscape analysis. *arXiv preprint arXiv:1806.06977*.

Kusano, G., Hiraoka, Y. e Fukumizu, K. (2016). Persistence weighted Gaussian kernel for topological data analysis. In *International Conference on Machine Learning (ICML)*.

Power, A., Burda, Y., Edwards, H., Babuschkin, I. e Misra, V. (2022). Grokking: Generalization beyond overfitting on small algorithmic datasets. In *ICLR 2022 Workshop on Trustworthy and Reliable Large-Scale Machine Learning Models*.

Singh, G., Mémoli, F. e Carlsson, G. (2007). Topological methods for the analysis of high dimensional data sets and 3D object recognition. In *Symposium on Point-Based Graphics*.

Zomorodian, A. e Carlsson, G. (2005). Computing persistent homology. *Discrete and Computational Geometry*, 33(2):249–274.

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945.

Graves, A. (2016). Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*.

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research (JMLR)*, 6:695–709.

Jacot, A., Gabriel, F. e Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31.

Kakade, S. e Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Maleki, S., Tran-Thanh, L., Hines, G., Rahwan, T. e Rogers, A. (2013). Bounding the estimation error of sampling-based Shapley value approximation with/without stratifying. *arXiv preprint arXiv:1306.4265*.

Montufar, G. F., Pascanu, R., Cho, K. e Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 27.

Ng, A. Y., Harada, D. e Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning (ICML)*.

Pearlmutter, B. A. (1994). Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160.

Pineau, J., Gordon, G. e Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. e Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Smallwood, R. D. e Sondik, E. J. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088.

Villani, C. (2003). *Topics in Optimal Transportation*. American Mathematical Society.

von Neumann, J. (1950). Functional operators. *Annals of Mathematics Studies*, Princeton University Press.